HAAGA-HELIA
University of Applied Sciences

Development and implementation of Finvoice in Microsoft Dynamics NAV 2016 to generate Finvoice 2.01 XML.

Oluwasegun Idris Ogidan

Bachelor's Thesis
Degree Program in Business Information Technology
20 May 2016

HAAGA-HELIA
University of Applied Sciences

Abstract

Date

Author(s)
Oluwasegun Idris Ogidan.

Degree program
Business Information Technology

Report/Thesis title	**Number of pages and appendix pages**
Development and implementation of Finvoice in Microsoft Dynamics NAV 2016 to generate Finvoice 2.01 XML file.	22 + 3

This project is based on the initiative of Tero Hassinen, a senior Microsoft Dynamics NAV Developer at Navpartneri Oy. It was intended to provide Finvoice option for clients using Microsoft Dynamics NAV as their enterprise resource planning (ERP) system. Navpartneri Oy is a small and fast growing company and a Microsoft Partner, who specializes in Microsoft Dynamics NAV ERP Implementations, Developments and customizations, Training, support services and so on.

The project was initially commissioned by Navpartneri Oy, but it ended as a personal project due to Tero's unexpected change of situation which we never envisaged before to starting the project.
The goal of this project is to be able to generate an XML file that conforms to the Finvoice 2.01 XML format and standard specified in the Finvoice Implementation Guide.

Finvoice is a product of the Federation of Finnish Financial Services (FFI), and it is an electronic invoice used by most companies and all banks in Finland.

Microsoft Dynamics NAV 2016 is an enterprise resource planning system for small and mid-sized companies, and it is owned by Microsoft.

I did my work placement with Navpartneri Oy and during that time, I was exposed to a few projects that they have already marked to be done. Implementation of Finvoice in Microsoft Dynamics NAV is one these projects.

To be able to successfully implement this project, a good understanding of Client Server Integrated Environment (C/SIDE), Client Application Language (C/AL), and Extensible Markup Language (XML) is needed. Prior experience working with Microsoft Dynamics NAV modules and basic functionalities is also needed as it is the foundation on which the project is built.

The Scope of project is to customize an already existing standard Microsoft Dynamics NAV 2016 application so that a Finvoice XML file is generated when a customer is invoiced. The out of scope of this project is the implementation of web services which will transport the generated XML file between the necessary parties such as the seller, buyer, and the bank.

Keywords
E-invoicing, ERP, Finvoice, Microsoft Dynamics NAV, Sales Invoice, XML.

Table of contents

Abbreviations

Terms	Definition
AP	Account Payable
AR	Account Receivable
C/AL	Client/Server Application Language
C#	C-Sharp Programming Language
C/SIDE	Client/Server Integrated Development Environment
CSV	Comma Separated Values
DLL	Dynamic- link Library
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
FFI	Federation of Finnish Financial Services
LCY	Local Currency
MSDN	Microsoft Developer Network
PDF	Portable Document Format
RTC	Role Tailored Client
SQL	Structured Query Language
VAT	Value Added Tax
WCF	Windows Communication Framework
XSD	XML Schema Definition
XML	Extensible Markup Language

1 Introduction

Automation of business processes is a feature that is widely embraced and adopted my most companies in Finland and globally. Finland is one of the most sophisticated countries as regards technologies and as a result, most companies in Finland have automated their invoicing processes through an electronic invoicing (e-invoicing) system to enhance the smooth running of their business dealings.
The use of electronic invoice is very popular among companies in Finland nowadays, and the main electronic invoicing system that is used is Finvoice.

Exchange of invoicing documents between the seller and the buyer in an integrated electronic format is known as electronic invoicing. (OpenText, 2016)

An Invoice that is generated through an electronic invoicing process is referred to as an electronic invoice.
The invoicing document that is sent from the seller to the buyer must be in a format understood by the buyer's application. Figure 1.1 below represents a visual definition of electronic invoicing:

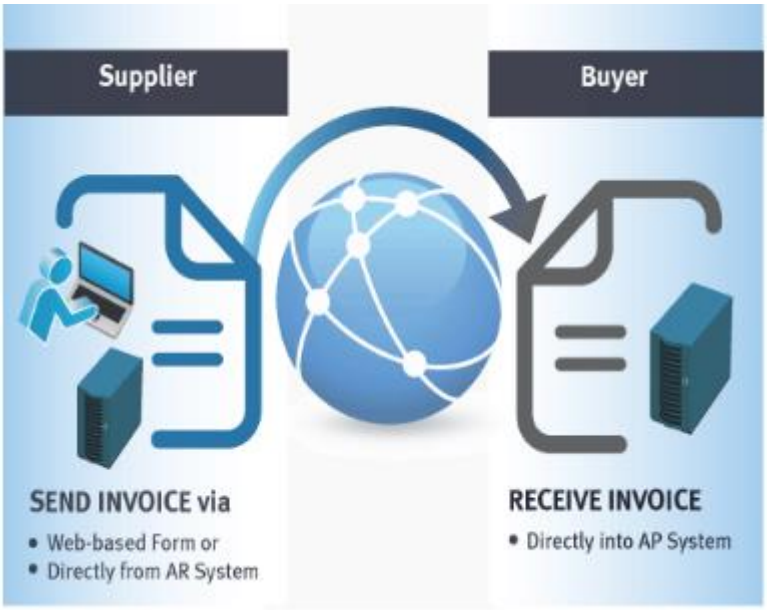

Figure 1-1 Electronic Invoice Definition

(OpenText 2016)

E-invoicing enables a company to automate its invoicing processes. As a result, the buyers and suppliers gain a number of operational and strategic benefits (OpenText 2015, 3a).

This is a product-based thesis and it based on the initiative of Tero Hassinen, a senior MS Dynamics NAV Developer at Navpartneri Oy and it is about the implementation of Finvoice in MS Dynamics NAV. Navpartneri Oy has a number of customers that have expressed interest in having Finvoice incorporated into their MS Dynamics NAV systems. This is the reason implementation of Finvoice into Microsoft Dynamics NAV was marked as one of the projects the company needs to do. In other words, the customers have asked for automation of their invoicing process.

Navpartneri Oy is a startup company and a Microsoft partner who specializes in Microsoft Dynamics NAV implementations, developments, support services and training.

1.1 Goal

The goal of this project is to modify the sales posting routines of Microsoft Dynamics NAV 2016 so that the system can generate a valid Finvoice 2.01 XML file when a sales order is posted, and a posted invoice is generated. The Finvoice XML that will be generated will be used for an electronic invoicing purpose and it must conform with the format and standard as in the Finvoice implementation guidelines.

To scope of this thesis involves customization of the sales posting routine of Microsoft Dynamics NAV system such that when a sales order is posted and a customer is invoiced, a Finvoice XML file is generated without disrupting the standard posting routines.
By standard, the posting routine Codeunit works in such a way that when a sales order, posted documents such as sales shipment and sales invoice documents.

Generally, Invoicing involves document exchange from the seller to the buy and for e-invoicing purpose. The document that will be sent from the seller will be the Finvoice XML file generated and this will be sent with the use of web services. The scope of this project is already large, and implementation of web services will be the out of scope of this project.

1.2 Structure of the thesis

This subchapter briefly describes the structure of this thesis. The first chapter represents the introduction and it briefly describes automation of invoicing processes (electronic invoicing), Finvoice, Microsoft Dynamics NAV and Navpartneri Oy.

The second chapter provides some theoretical background information that supports the actual development work as this is a product-based thesis.
This will be grouped into different parts. In the first part, electronic invoicing concepts and Finvoice as an electronic invoicing system used in Finland will be discussed.
There will be an overview of the Microsoft Dynamics NAV 2016 in chapter two. This is the ERP system on which the product is built. Extensible Markup Language (XML) as well a brief on previous work done in a similar industry will also be discussed in this chapter.

The third chapter basically discusses the development plan, how the project will be implemented. Software development method used and all the tasks to be done will be discussed.

The fourth chapter explains the actual development process from start to finish.

Chapter 5 describes the result of the development work done and how the result will be tested and evaluated.

2 Theoretical Framework

Finvoice implementation in Microsoft Dynamics NAV 2016 in order to generate Finvoice XML file is a project which involves different technologies, concepts and applications.
Microsoft Dynamics NAV 2016 is the ERP system on which the customization is done and it is owned and managed by Microsoft Corporation.
The result of this project will be the Finvoice XML document that will be generated from Microsoft Dynamics NAV 2016. Finvoice is an XML representation of a business document and it is used for an electronic invoicing purpose.

The electronic invoicing concept has been in existence for many years, and acceptance was very slow then. Electronic data interchange (EDI) was used to send the first electronic invoices over thirty years ago. (OpenText 2016b)

Interest in electronic invoicing was very low during the early days because of the high cost of implementing it but over the years, there have been increased interest percentage from corporate accounting firms and institutions. The cost of implementing EDI then was very high, and it requires a lot of resources is the reason there was reduced interest in electronic invoicing implementation. That has changed a lot now due to improved technology.

Rochelle P. defined EDI as the computer-to-computer exchange of business documents, like invoices and purchase orders, between two business partners in a standard electronic format.

The figure below represents what a typical EDI system looks like:

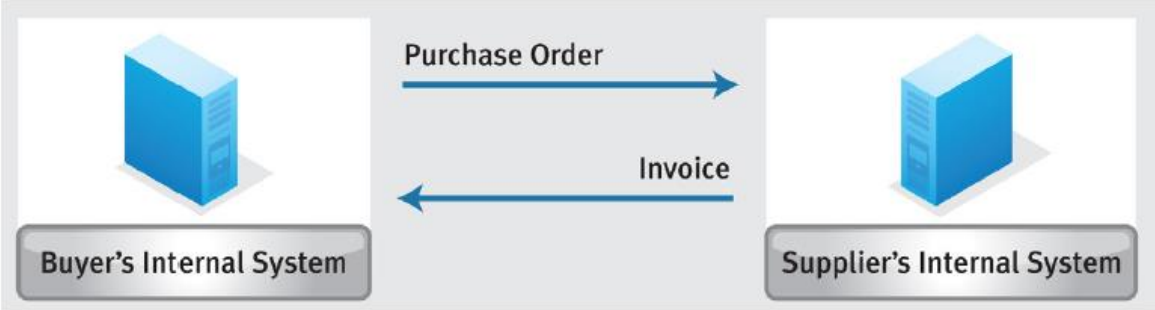

Figure 2-1 EDI Document Exchange

(Rochelle. P)

Nowadays, electronic invoicing is much more affordable because there are varieties of data exchange medium and a very improved and efficient internet to transport data.
In Europe, e-invoicing is mostly controlled by the government and some regulatory bodies. The Federation of Finnish Financial Services (FFI) controls and manages electronic invoicing system in Finland.

There are different types of e-invoice and as a result, it is difficult to know what the true e-invoice is.
In the past, invoices that were sent from the sellers to the buyers in form of a scanned document and in a PDF format have also been considered as e-invoicing but these types of invoicing approaches are not considered true e-invoicing because the receiver of the invoice still has to do some manual works by first receiving the invoice, processing it and store it.

This additional manual work is eliminated in the true e-invoicing completely because “the buyer receives invoices from its supplier in a format that both parties had already agreed earlier.” (OpenText 2016c)
The format could be in XML or any other format understood by the buyer’s enterprise resource planning system.

A true e-invoicing is either compliance-dependent or compliance-independent.

Compliance-independent E-invoicing

Compliance-independent e-invoicing is an e-invoicing process that does not depend on compliance, and it works like this:

- The format of the e-invoice and the type of data it will contain will be agreed between seller and the buyer.
- Technologies like EDI or a web form will be used to structure the data.
- The seller generates the e-invoice from the Accounts Receivables (AR) system in the agreed format and sends it to the buyer.
- The buyer receives the e-invoice into the Accounts Payable (AP) system in a format the AP understands. (OpenText 2016c)

Compliance-dependent E-invoicing

A compliance-dependent e-invoicing involves the following processes in addition to the processes involved in a compliance-independent e-invoicing:

- The legislative and regulative body of the territories in which the seller and the buyers trade determines the format of the e-invoice.
- Invoices follow guidelines such as content validation, non-repudiation, security and archiving specified by the government on e-invoice.
- For audit and VAT purposes, the invoices are made available.
- It is completely paperless.

The buyer receives the electronic invoice in a format ready to be processed by the Account Payable (AP) system. (OpenText 2016c)

2.1 Finvoice Overview

Finvoice is an e-invoicing system used in Finland and it is owned and maintained by the Finnish Financial Federation Services (FFI). It is defined using XML syntax. The Finvoice messages can be used for invoicing and other business messages, such as quotations, orders, order confirmations, price lists, etc. (FFI 2015, 4). Finvoice has been in existence since 2003 and there have been different versions as shown in the table below:

Finvoice Versions	Year
1.0	2003
1.1	2004
1.2	2005
1.3	2008
2.0	2012
2.01	

Table 2-1 Finvoice Versions

To use the Finvoice financial services, both the sender and the receiver of are required to have signed the Finvoice forwarding agreement with their service providers. (FFI 2015, 5)

The service provider sends the e-invoice to the online bank or web payment service of the consumer if the consumer has already agreed.

The seller generates an invoice document and sends it to the buyer through its service provider.

The buyer receives the invoice document to buyer's system or views it in the browser.

Tieke website provides the list of the company names, sender's identifiers, receiver's identifiers and codes for service providers.

Finvoice messages are used for sending business messages like quotations, orders, invoices, and so on. It is popularly used for sending an invoice and that is what the focus of this project is on.

At the very list, a Finvoice message the following very mandatory data:

- Seller's Details
- Buyer's Details
- Invoice Detail
- Invoice Row Details
- Payment Information (ePI). (FFI, 2015)

The Message Transmission Details is a mandatory part of the Finvoice message structure, and it contains message details accompanying the invoice, the details of the message sender and the message receiver.

Seller's details contain information about the seller. Seller's names, addresses, business ID, website, bank details and so on are some of the information under seller's details.

Buyer's details contain information about the buyer. Buyer's names, addresses, business ID, website, bank details and so on are some of the information under seller's details.
Invoice details contain information like buyer's details, invoice number, invoice date, invoice discount, VAT and so on.
The invoice row details contain information about the items being sold such as item number, quantity, unit price, amount and so on.
Payment Information (ePI) contains payment information from the seller.

2.2 Microsoft Dynamics NAV as an Enterprise Resource Planning

In every business organization, there are different departments and functions and before the existence of enterprise resource planning (ERP) system, one of the challenges faced by organizations is how to gain timely access to information. Since organizations are made up of several functions, it means that the organization will have information scattered across all business functions. Each business function will each have a database on which information is stored, and all the databases may be operating independently from one another.
Databases are referred to as island systems. (Microsoft 2015a, 2)

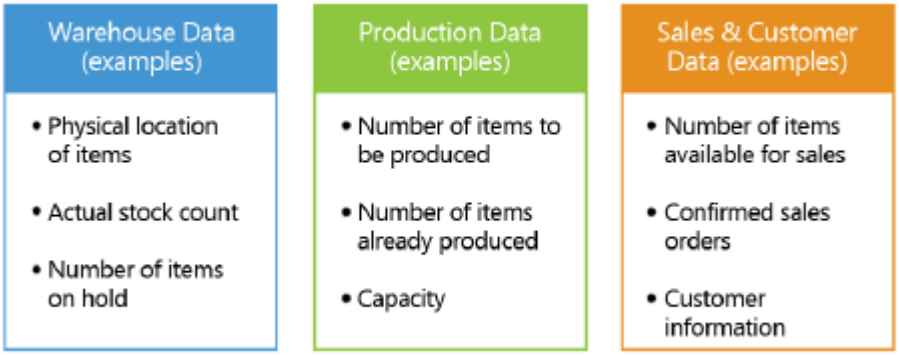

Figure 2-2 Island Systems (Microsoft 2015a, 2)

Figure 2.2 above represents island systems.
Having the warehouse information, production information, sales and customers' information in different databases across the enterprise and with no integration makes it difficult to retrieve information.
The limitation with Island Systems which is as a result of having information in several databases across the enterprise is completely eliminated in ERP systems because all information about each business function is available in one single database.

Figure 2.3 below represents what a typical ERP system looks like:

Figure 2-3 ERP Systems (Microsoft 2015a, 3)

The information is available real-time when using ERP Systems, and this is not the case with legacy systems.
ERP systems can be modified to suit different company needs as it is with this project where Microsoft Dynamics NAV 2016 will be customized to generate Finvoice XML file after a customer is invoiced. This is however not part of the standard features of Microsoft Dynamics NAV.

> Microsoft Dynamics NAV 2016 is a fully integrated relational database system which allows organizations to create and manage financial and other business processes in a single platform. It contains financial management, sales and receivables, inventory, purchases and payables, jobs, service management, human resource management, manufacturing, warehouse management and resource management.
> These features make Microsoft Dynamics NAV 2016 an ERP system. (Microsoft 2015a, 5)
>
> Microsoft Dynamics NAV 2016 is an enterprise resource planning and a business management solution designed for small and mid-sized organizations to help streamline and automate business processes. (MSDN 2016a)"
> One interesting feature of Microsoft Dynamics NAV 2016 is that it is customizable. Developers can modify some existing features to suit different customer needs.

2.3 Microsoft Dynamics NAV 2016 Architecture

Microsoft Dynamics NAV 2016 is built on a three-tier architecture, and the 3-tier architecture of Microsoft Dynamics NAV is made up of the client tier, server tier and service tier.

Client Tier
This is the Role tailored client which interfaces with the users. It exists as a Microsoft Dynamics NAV Windows client and as a Microsoft Dynamics NAV Web client. Microsoft Dynamics NAV also supports other clients like Web Service client.
The role tailored client is shown in the figure below:

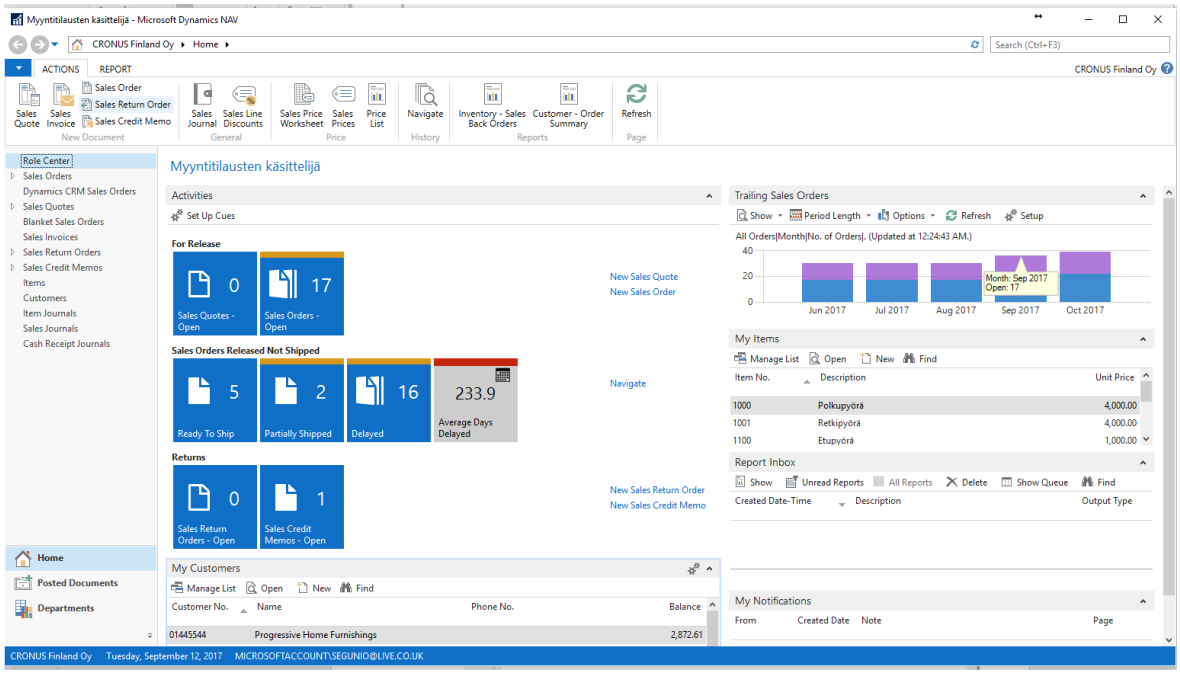

Figure 2-4 Microsoft Dynamics NAV 2016 Role Tailored Client

Server Tier

The server tier of the Microsoft Dynamics NAV 2016 architecture is a .NET-based Windows service application. It is the Microsoft Dynamics NAV Server which manages all business logic and communication. Between Microsoft Dynamics clients and Microsoft Dynamics databases. It uses a communication protocol known as Windows Communication Framework (WCF), and It is also referred to as the middle tier.

Data Tier

The data tier in the Microsoft Dynamics NAV 2016 architecture is made up of the computer which runs the SQL Server database with which Microsoft Dynamics NAV interacts. This is the SQL Server which houses and manages the Microsoft Dynamics NAV database. It contains the SQL Server database components which configure Microsoft SQL Server to work with Microsoft Dynamics NAV 2016.

SQL Server Express edition comes with the Microsoft Dynamics NAV installation package and it is installed when installing Microsoft Dynamics NAV. If another SQL Server version such as Enterprise or Standard edition has already been installed on the computer, the installation configures Microsoft Dynamics NAV to with that version.

In a production environment, configuring Microsoft Dynamics NAV to work with an Express edition of SQL Server is not recommended.

The figure below represents a Microsoft Dynamics NAV 2016 architecture:

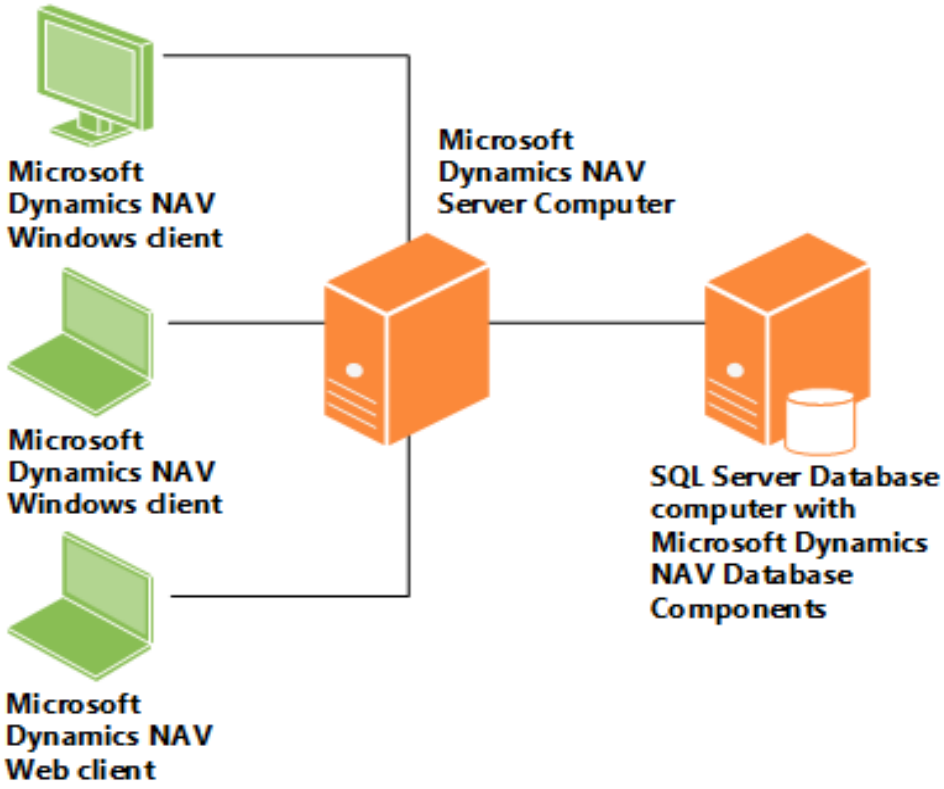

Figure 2-5 Microsoft Dynamics NAV 2016 Architecture (MSDN 2016b)

There are three common types of configuration for Microsoft Dynamics architecture in respect to the number computers used to represent the architecture.
For the three-tier architecture on three computers, the client tier, the server tier, and the data tier are each installed on different computers.

For the three-tier architecture on two computers, the client tier and the server tier run on the same computer while the data tier runs on the second computer.

The three-tier architecture on a single computer is the third option where the client tier, the server tier, and the data tier are installed on the same computer. This is the installation option for a demo installation.

The three-tier architecture on a single computer is the configuration type used for this project. “It is typical for a development environment which developers use to develop Microsoft Dynamics NAV applications” (MSDN 2016b).

2.4 Microsoft Dynamics NAV Development Environment

Microsoft Dynamics NAV is the main application on which this project is developed, and all development works are done in C/SIDE.
C/SIDE which stands for Client/Server Integrated Development environment is the development environment for MS Dynamics NAV and Client/Server Application Language C/AL is the programming language used for creating new and modifying existing objects in C/SIDE. It is very similar to Pascal programming language.

C/SIDE Objects
Microsoft Dynamics NAV C/SIDE is an object-based application and as a result, C/AL programming is an object-based programming.
The C/SIDE application is made up of eight different objects, and they are encompassed in the Object Designer. Creation of new objects and modification of already existing objects are done in the object designer.

The C/SIDE application objects are shown in the table below:

Application Objects	Definitions
Table	Tables are used to store data.
Form	Forms are used to view and present data stored in tables in the Microsoft Dynamics NAV classic client. The Microsoft Dynamics NAV classic client is no longer available in Microsoft Dynamics NAV 2016. Forms are the user interface for creating new and modifying already existing data in tables e.g. customer card, quote document, order document, invoice document and so on.
Report	Reports are object designers that are used to present data from tables in a printable form e.g. Bank Statement, Trial Balance, etc.
Dataport	Dataports are object designers that used for importing data and exporting data in text file formats e.g. Excel formats or CSV.
XMLport	XMLports are object designers that are used for importing data and exporting data in XML formats.
Codeunit	Codeunits contains user-defined functions written in C/AL codes and these functions can be referenced or called from other objects in the C/SIDE application.
MenuSuite	MenuSuites are used to design the menus displayed in the navigation pane.
Page	Pages are used in place of forms in the Role Tailored client (RTC)

Table 2-2 C/SIDE Application Objects

The application objects that will be used for this project are Tables, Pages, XMLports, and Codeunits.

The object designer is used to design new object and modify already existing objects. Object designers are used design object type. In other words, this means that a table is designed by the table designer, a form is designed using form designer and so forth in that order. The figure below shows the object designer in the Microsoft Dynamics C/SIDE.

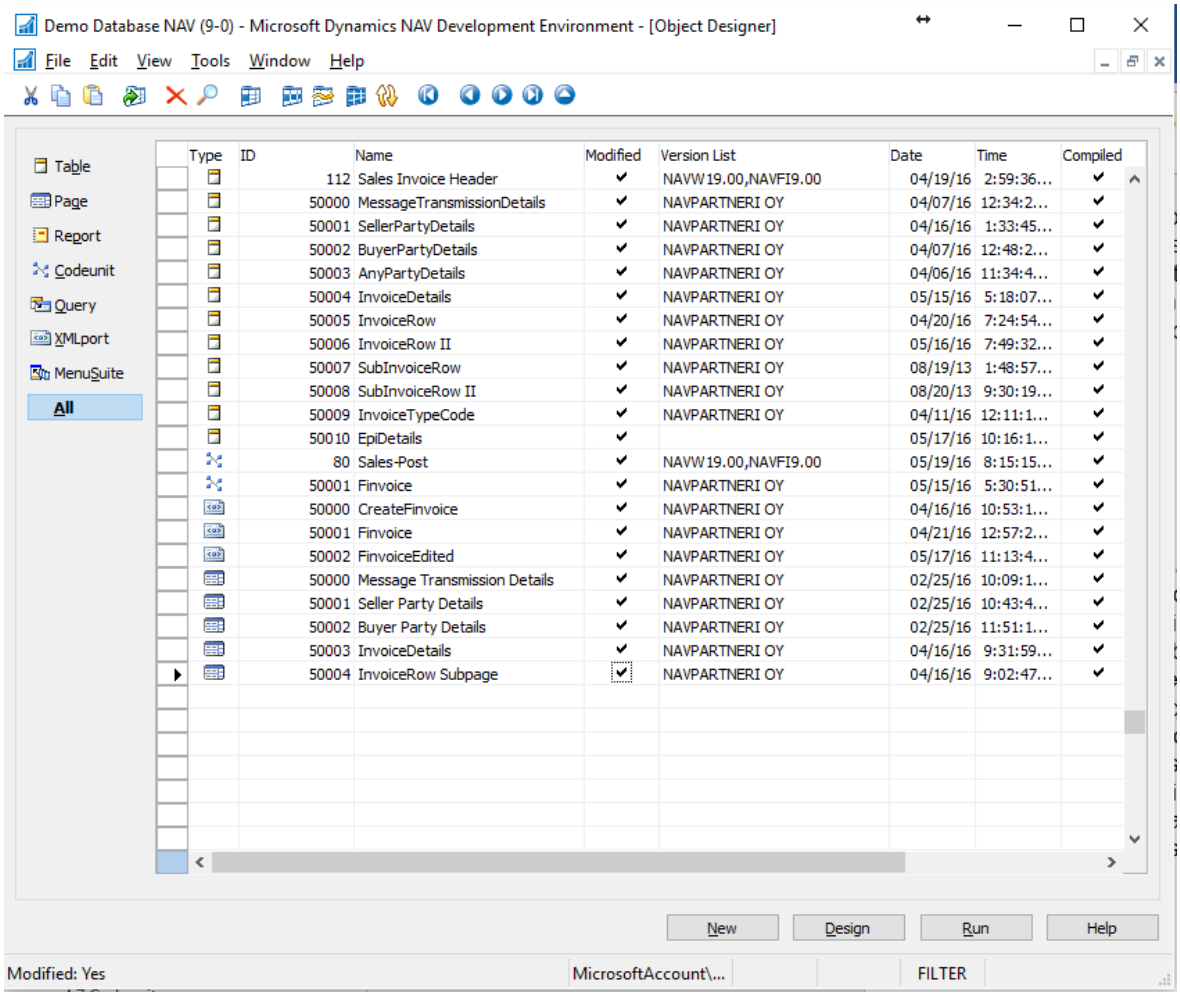

Figure 2-6 Microsoft Dynamics NAV 2016 Object Designer

2.5 Extensible Markup Language

“XML is a tool for storing and transporting data and it does not depend on software and hardware.” (W3school 2016)
Data transported between different applications are mostly sent in XML format because it is a format that is understood by almost all applications. XML was designed to be readable by human beings too. An XML is made up of elements and each element describes and holds data.

The final result of this thesis work is to generate an XML file which conforms to the format and standard specified in the Finvoice implementation guidelines. To test this conformity, the generated XML file will be validated with the Finvoice 2.01 Schema.
The Finvoice schema is an XML schema definition (XSD) that was designed by the FFI for validating Finvoice XML files.
In this project, the XML file will be generated from Microsoft Dynamics NAV and XML files can be generated from Microsoft Dynamics NAV 2016 with the use of XMLports and Codeunits. XMLports and Codeunits will be discussed later.

Microsoft Visual Studio is one of many applications that is used to read and edit XML documents. It will be used to read and edit the XML file that will be generated.

2.6 Related Project

A similar project has been done by Qin Jin, a graduate of Oulu University of Applied Sciences. Qin's project was titled 'Finvoice Generating' was a commissioned thesis offered by Jukka Penttilä from Sunwell Trade. The company has a billing system called REX, and they need a Dynamic-link library (DLL) that can be used to generate valid Finvoice documents. This project involved building the required Dynamic-link library.
For successful completion of this project, prior knowledge of C# programming and Extensible Markup Language (XML) was required and the development tool used was Visual Studio 2010. Notepads were also used to display and edit the generated XML documents.
Qin also developed an application in C# to test the DLL.

3 Development Plan

In this project, Microsoft Dynamics NAV 2016 will be customized to be able to generate Finvoice 2.01 XML file as part of the actions carried out when posting sales order. Most of the development work will be done on C/SIDE, which is the integrated development environment for developing new objects and modifying already objects in Microsoft Dynamics NAV.

3.1 Project Work Breakdown

This project like any normal project involves several tasks and they are listed below:

- Install SQL Server, which will house and manage the database.
- Install and configure Microsoft Dynamic NAV Client and Server with a demo Cronus database.
- Upload Developer License.
- Create the Finvoice Tables.
- Create pages to display the Some Finvoice data in the form of a Sales document with a header and lines.
- Modify Sales-Post Codeunit, which controls the posting routines of sales documents to update the Finvoice tables with the data from the posted documents generated and from some other necessary tables that are not part of the original invoice.
- Create the XMLport that will be used to input to and export from the Finvoice tables.
- Create a Codeunit to run the XMLport to generate the Finvoice XML file.
- Testing System.
- Validate the Finvoice XML file.

3.2 Development Method

Software projects are managed using different software development methods. Examples are Scrum, Waterfall model, Rapid Application Development and a host of others.
However, the development method that will be used for this project is the waterfall model because of its flexibility.

3.3 Standard Microsoft Dynamics NAV Sales Invoice Process

Often, a typical sales process starts when a customer requests for a quotation on some items and sometimes, the sales process starts when a customer places an order for one or more items. The figure below shows a typical sales process flow and how it affects the inventory and the General Ledger.

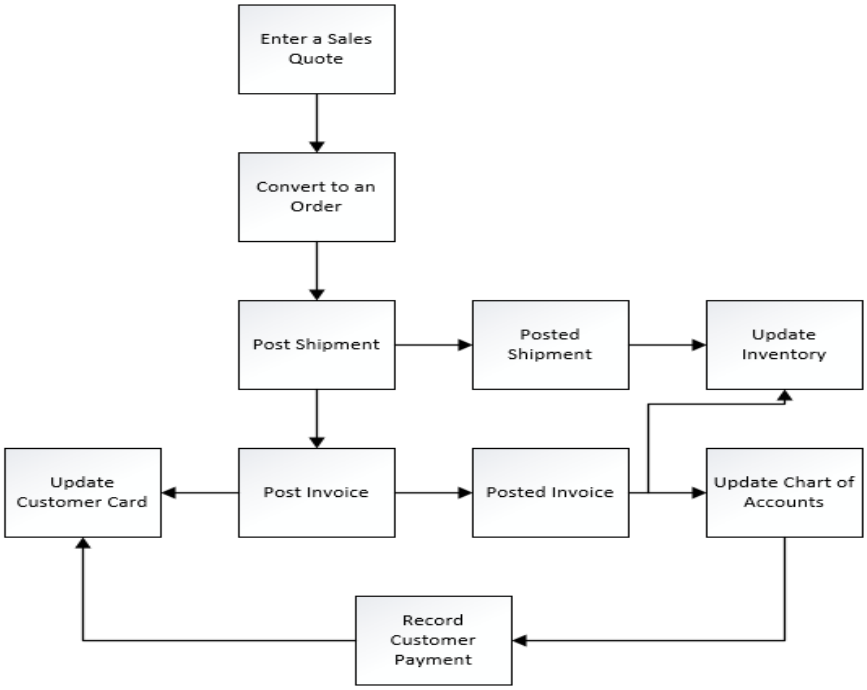

Figure 3-1 Sales Process without Finvoice

In Microsoft Dynamics NAV, a sales quote is created when a customer or a prospective customer needs for a quotation to know the prices and availability of an item or a set of products.

The sales representative creates a sales quote, and sends to the customer. When the customer receives it, she/he makes a decision whether to purchase the product(s) or not. If the customer decides to purchase the product(s), the Sales representative converts the sales quote to a sales order. To convert a quote to an order, the sales representative will click the 'make order' and this action will delete the sales quote and create a new sales order. The sales quote number is available on the sales order for reference purpose.

The sales quotes are deleted to save database space but the decision to delete a sales quote when converting them to a sales order can be controlled as it is setup dependent.

After confirmation, the sales representative posts shipment to generate a posted shipment and update the inventory.

Inventory is the number of products in the store. The inventory is decremented by the quantity on sales line during sales and incremented by the quantity on the purchase line during purchase.

After that, the sales representative posts invoice. This action generates posted invoice, updates inventory, customer account and chart of accounts.

When the customer makes payment, the system updates the customer's account and

After the implementation of Finvoice, the sales process flow looks like this:

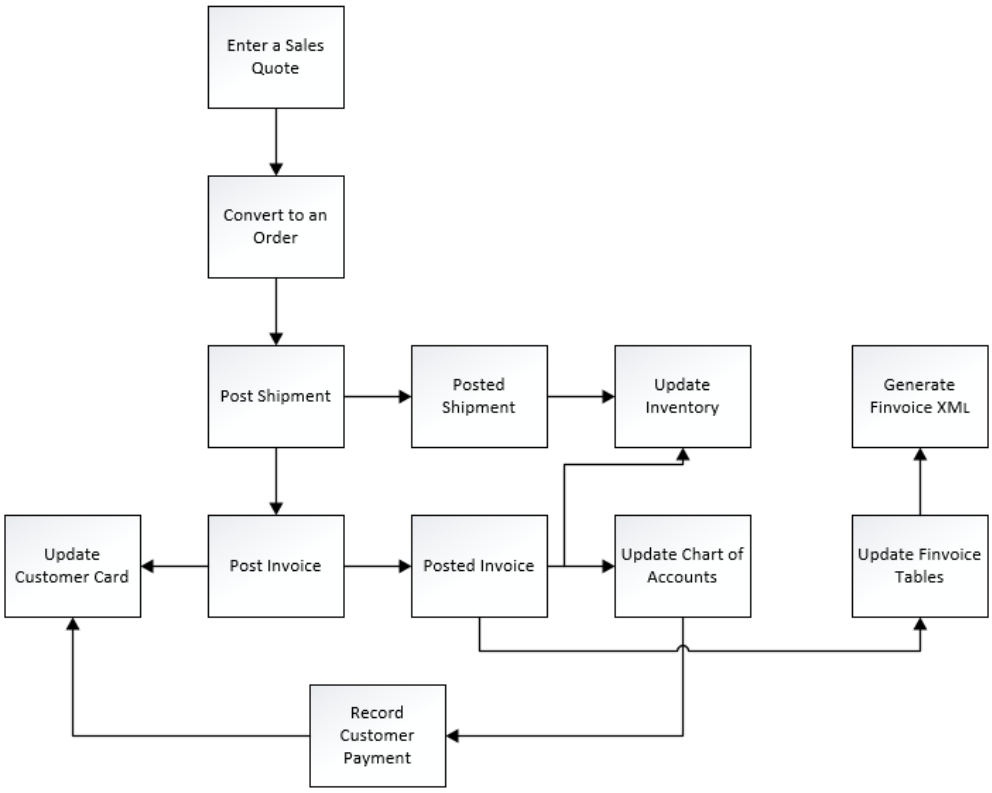

Figure 3-2 Sales Process with Finvoice

This means that the standard sales process remain as it is even after implementing Finvoice. The only difference is that the system performs additional actions. After implementation of Finvoice, the System works in such a way that when a posted invoice is created, the system updates the necessary Finvoice tables and generates a Finvoice XML file.

4 Development and Implementation

This chapter explains the actual development work done from the installation and configuration of Microsoft Dynamics NAV 2016,

4.1 Installation and Configuration

Microsoft Dynamics NAV 2016 and its prerequisites are essential components that need to be installed because it is the means of gaining access to the application and its development environment.
The demo installation option of MS Dynamics NAV 2016 would have been good enough for this project but I could install it successfully with the demo option as a result of the fact that I had already installed SQL Server 2016 as one of the prerequisites. Normally when a demo version is chosen during installation, the system installs SQL Server Express Edition as well as other prerequisites that come with the MS Dynamics NAV installation package. I installed Microsoft Dynamics NAV 2016 with a custom configuration.

4.2 Developer License

Microsoft Dynamics NAV is a Licensed based application which is based on a concurrent number of users. Microsoft Dynamics NAV license capabilities vary depending on the type of license a user has. For instance, buying a license with twenty simultaneous users imply that only twenty users can use the system at the same time, but the number of users that can have access to use the system can be more than twenty.
Most Microsoft Dynamics NAV customers use the normal license while vendors or service provider and developers use the developer license.
Microsoft Dynamics NAV licenses are neither forward compatible nor backward compatible and this implies that the Microsoft Dynamics NAV 2016 cannot work with earlier versions of Microsoft Dynamics NAV and vice versa.
When a demo version of Microsoft Dynamics NAV is installed, it comes with a demo license and this demo license provides users with read, write, delete rights to some records. The demo license does not give users access to use the object designer and users with demo license will also be unable to run some objects that the license does not have access to run. In a real life implementation for customers, it is highly recommended to create a new Microsoft Dynamics Database and to create a new database and a new company because a user needs a license with such capability. Another limitation of using a demo license is that a user cannot create a new database with a demo version.
Companies using Microsoft Dynamics NAV as their ERP system use the customer license type which obviously have more rights. A customer license may include some development rights depending on the level of competence of some of their users.
A developer license is required to be able to use the object designer for creating new objects and modifying existing objects. For this project, I was given a developer license to work with.
According to (MSDN 2016e), the types of Microsoft Dynamics NAV License into a full user, limited user, device only user, Window Group and external user.
A full user license is a license which gives users full read and full write rights.
A limited user license is a license which gives users full read but limited write rights.
A device only user license can be either limited or full. It is a license which allows a user to connect with available devices.
A Windows group license is a license that is used by users under Windows group. Each user inherits the permissions of the Windows group.
An External User is a license for specific users.

When a developer visits a Customer's office and needs to do a quick modification, she/he uses the change license option in the License Information window and the license is only saved on that instance of Microsoft Dynamics NAV. The upload option is used to replace an existing license.

To change or upload a license on License Information window, follow the following steps:

- Got to Tools
- Click on License Information

Choose change or upload as the case may be. (MSDN 2016c)

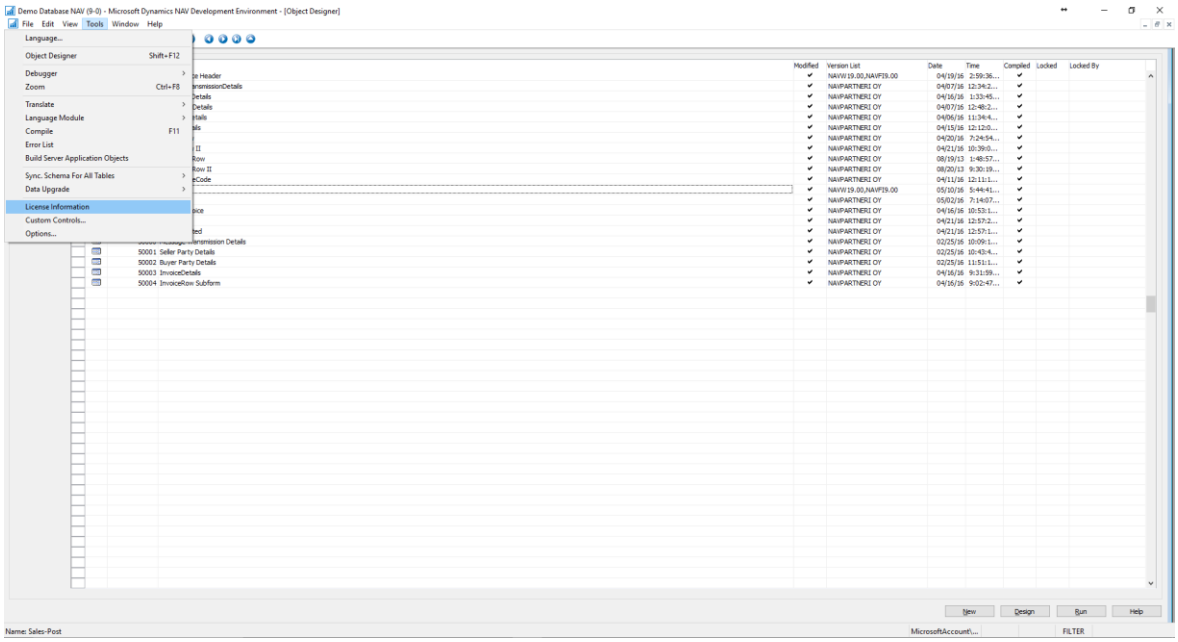

Figure 4-1 How to view License Information

Modifying an existing object irrespective of the type starts with the same procedure as listed below:

1. Click on start menu and click on the Microsoft Dynamics NAV 2016 Development environment. This should open object designer as shown below:

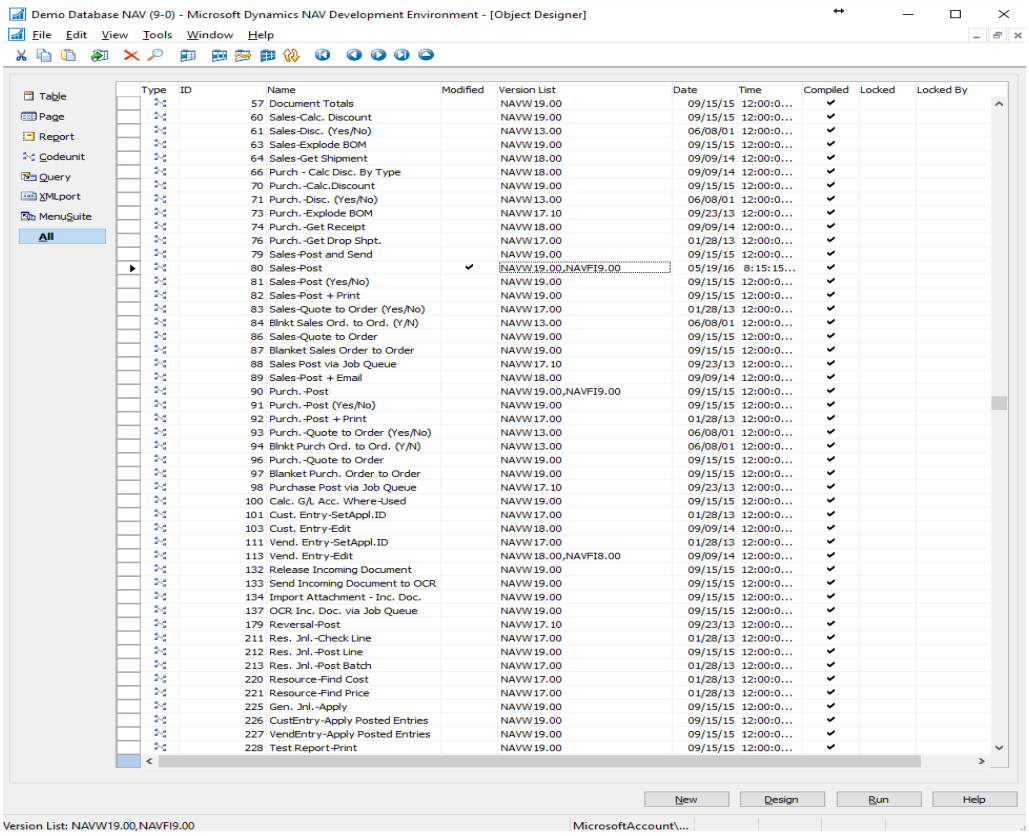

Figure 4-2 Object Designer showing All Object Types

2. Select the object type. The figure above shows all the object types. To design a Table for instance, click on Table object and this will show the list of tables. To modify an existing table, search the name of the table, select it and click the design button below. This will open the table in design mode and the content can

be viewed or edited. See the figure below:

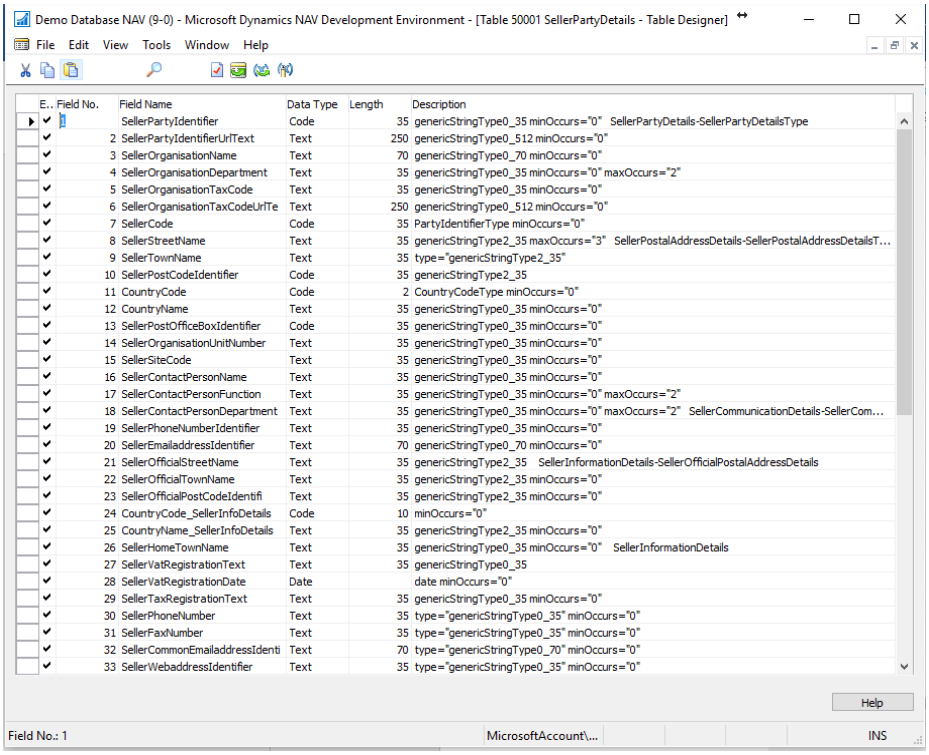

Figure 4-3 Table Designer SellerPartyDetails

If the new button is clicked, a new table designer opens as shown below:

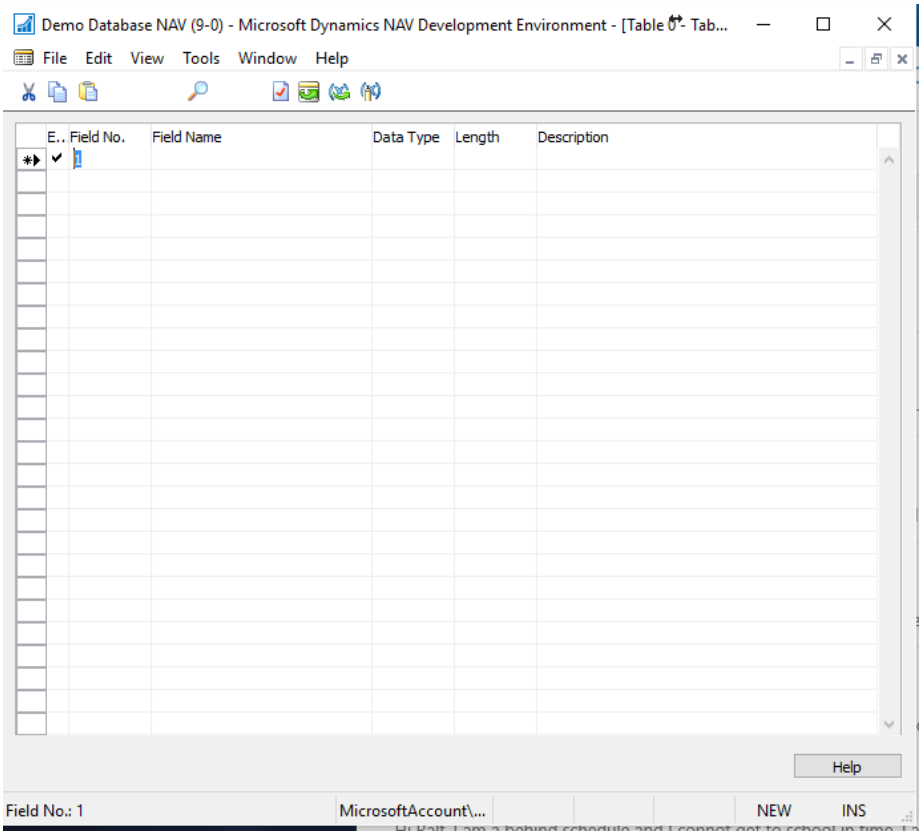

Figure 4-4 Table Designer for new Table

3. Modify the table. For example, edit a new field or create a new field.

4. Write some C/AL codes to change the default behavior of the fields or to carry out some actions.
5. Save the changes made and compile the table. An object will only compile when it is error free.

4.3 Finvoice Tables

Initially, I did not plan to create Finvoice tables because the system already has tables containing information equivalent to the information in the Finvoice tables but I decided to create new tables for Finvoice due to the fact that the fields from the Finvoice table are unique in terms or their naming system.
Meanwhile, I wanted to avoid tampering with the standard Microsoft Dynamics NAV standard tables to avoid having conflicts during deployment.
The table below shows the list of tables modified for this project. Sales Invoice Header is the only standard table modified, and all others are new tables.

Object Type	Object Name	New / Standard
Table	Sales Invoice Header	Standard
Table	MessageTransmissionDetails	New
Table	SellerPartyDetails	New
Table	BuyerPartyDetails	New
Table	AnyPartyDetails	New
Table	InvoiceDetails	New
Table	InvoiceRow	New
Table	InvoiceRow II	New
Table	SubInvoiceRow	New
Table	SubInvoiceRow II	New
Table	InvoiceTypeCode	New
Table	EpiDetails	New

Table 4-1 FInvoice Tables

I created some functions in the Sales-Post Codeunit to update the Finvoice tables.
The functions will be explained under Codeunits.

4.4 Finvoice Pages

I created some pages to view the data in the Finvoice tables. The most important of those pages is the InvoiceDetails Page.

Object Type	Object Name	New / Standard
Page	Message Transmission Details	New
Page	Seller Party Details	New
Page	Buyer Party Details	New
Page	InvoiceDetails	New
Page	InvoiceRow Subform	New

Table 4-2 Finvoice Pages

The InvoiceDetails page was designed as a document type page which looks like a typical invoice document as shown in the figure below:

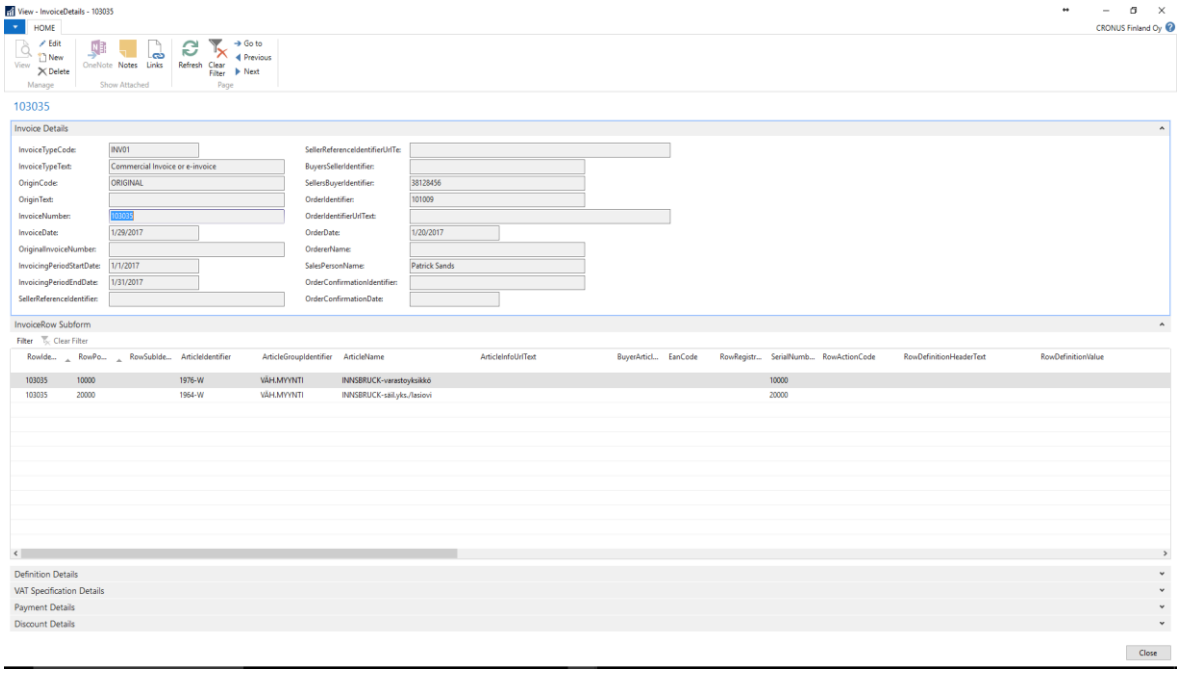

Figure 4-5 Finvoice Invoice Page

The page above is a document page which is a combination of the Invoice Details Page and the InvoiceRow Subpage.
The InvoiceDetails Page uses the InvoiceDetails table as its source-table from which it retrieves the information it displays while the InvoiceRow Subpage uses the InvoiceRow table as its source-table.

4.5 XMLport

XMLports is an object designer that is used to import data to Microsoft Dynamics NAV database in XML format and to export data from Microsoft Dynamics NAV database in XML format. (MSDN 2015f)
The Finvoice XMLport was designed to generate the Finvoice XML file. It is a complex XMLport because it uses multiple source tables and in some of the XML

elements were represented by variables. Due to the limitations on the size of fields in a table used in Microsoft Dynamics NAV, which cannot be more than 8000 bytes. This limitation made me create an additional table for the InvoiceRow, and I named it InvoiceRow II.
When creating the XMLport, each field is indented under their source table. For Instance, if InvoiceRow table uses indentation 1, it means all other fields under the InvoiceRow source table must use at least indentation 2.
The fields from InvoiceRow and InvoiceRow II table are supposed to come from one table because they are expected to be indented under the InvoiceRow table only and there is no InvoiceRow II in the Finvoice Implementation Guide.
This limitation means that some XMLport element will be represented as a variable and updated programmatically under the Export: OnBeforeAssgnedField () trigger.

Object Type	Object Name	New / Standard
XMLport	Finvoice	New

Table 4-3 Finvoice XMLport

The running of the Finvoice XMLport which generated the XML file is initiated from the Finvoice Codeunit.

The XML port in the design mode looks like this:

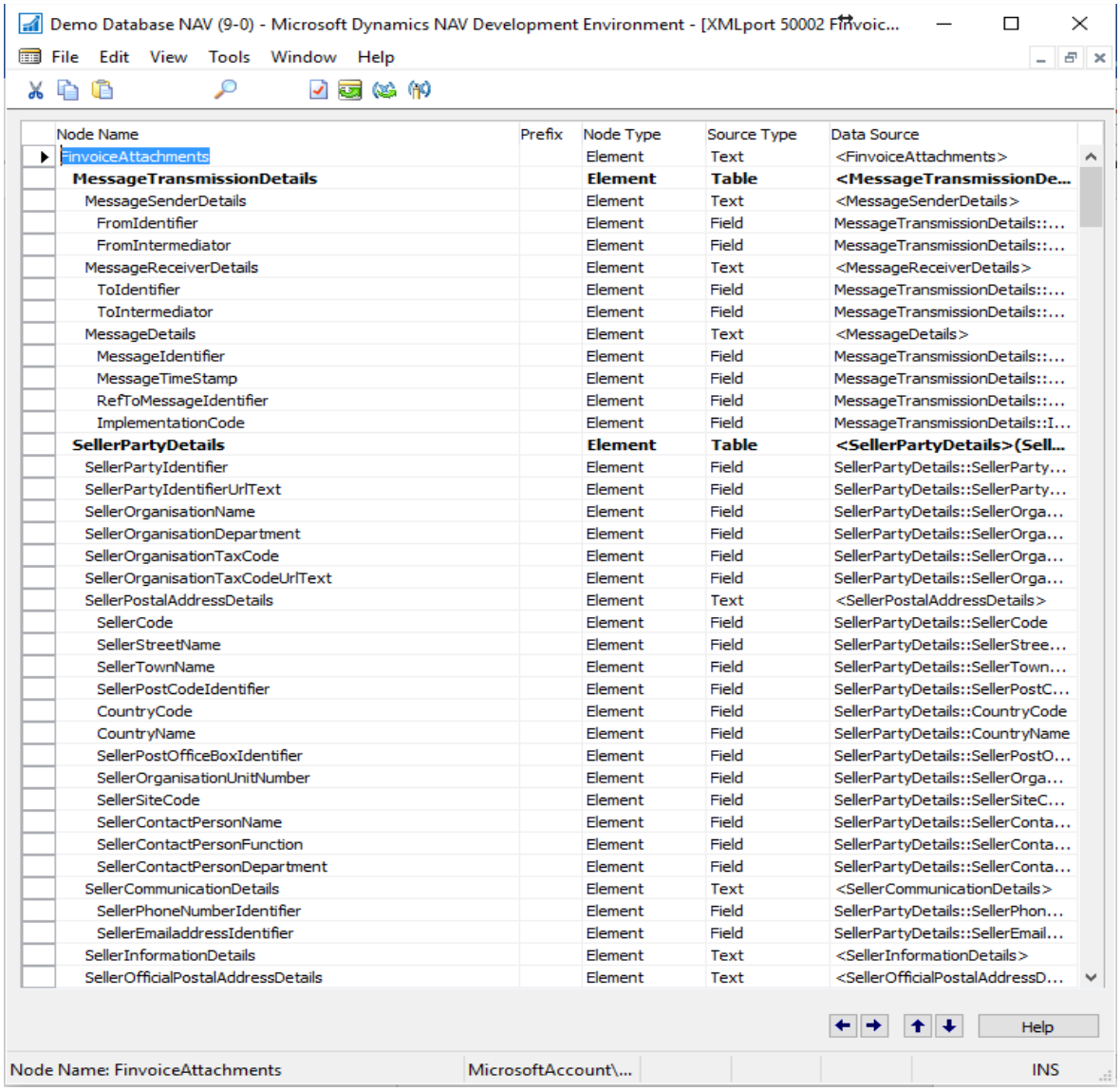

Figure 4.6 Finvoice XMLport in Design Mode

4.6 Codeunits

There two Codeunits used for this project, and they are:

Object Type	Object Name	New / Standard
Codeunit	Sales-Post	Standard
Codeunit	Finvoice	New

Finvoice Codeunit is referenced from Sales-Post after the main Finvoice functions have been successfully executed. This Codeunit calls and runs the Finvoice XMLport which generates the Finvoice XML File.
Sales-Post Codeunit is a standard Microsoft Dynamics NAV Codeunit which controls all the sales-related posting routines. The Sales-Post Codeunit contains the functions updating the Finvoice table. The functions are InsertMessageTransmissionDetails, InsertSellerPartyDetails, InsertBuyerPartyDetails, InsertInvoiceDetails, and InsertInvoiceRow.

InsertMessageTransmissionDetails

The InsertMessageTransmissionDetails function updates the MessagaTransmissionDetails table by fetching data from the following standard Microsoft Dynamics NAV tables:

- Company Information
- Customer Bank Account
- Sales Invoice Header

```
5183
5184 LOCAL InsertMessageTransmissionDetails(CompanyInfo : Record "Company Information";Cust : Record Customer;VAR MessageTransmissionDetails : Record MessageTransmissionDetails)
5185 WITH MessageTransmissionDetails DO BEGIN
5186   MessageTransmissionDetails.INIT;
5187   MessageTransmissionDetails.InvoiceNumber := SalesInvHeader."No.";
5188   MessageTransmissionDetails.FromIdentifier := CompanyInfo."Business Identity Code";
5189   MessageTransmissionDetails.FromIntermediator := CompanyInfo."SWIFT Code";
5190   MessageTransmissionDetails.ToIdentifier := Cust."No."; //To be continued!
5191   CustBankAcct.SETRANGE("Customer No.",Cust."No.");
5192   IF CustBankAcct.FINDFIRST THEN BEGIN REPEAT
5193     IF Cust.Name = CustBankAcct.Name THEN
5194       MessageTransmissionDetails.ToIntermediator :=CustBankAcct."SWIFT Code";
5195     UNTIL CustBankAcct.NEXT = 0;
5196   END;
5197   MessageTransmissionDetails.ImplementationCode := '';
5198   MessageTransmissionDetails.MessageIdentifier := 'MsgId' + CompanyInfo."Business Identity Code";
5199   MessageTransmissionDetails.RefToMessageIdentifier := ''; //To be reviewed later
5200   MessageTransmissionDetails.MessageTimeStamp := CURRENTDATETIME;
5201   MessageTransmissionDetails.INSERT;
5202 END;
```

InsertSellerPartyDetails

The InsertSellerPartyDetails function fetches data from the Company Information table and updates the SellerPartyDetails table with corresponding data. The SellerPartyDetails

table is one of the source tables for the Finvoice XMLport that will be used to generate the Finvoice XML file.

InsertBuyerDetails

The InsertBuyerDetails function fetches data from the customer tables and updates the BuyerPartyDetails table. The BuyerPartyDetails table is one of the source tables for the Finvoice XMLport and the data from this table will also be used for generating the Finvoice XML file.

InsertInvoiceDetails

The InsertInvoiceDetails functions updates the InvoiceDetails table with data from the standard table named Sales Invoice Header. The InvoiceDetails table is also one of the source tables for the XMLport that will be used for generating the Finvoice XML file

InsertInvoiceRow

The InsertInvoiceRow function updates the InvoiceRow table with the corresponding data from the standard table named Sales Invoice Line. One of the source tables for the XMLport that will be used to generate the Finvoice XML file is the InvoiceRow table.

Finvoice Codeunit

The Finvoice Codeunit creates a temporary path to store the XML file. It creates an instance of the XML export function called CREATEOUTSTREAM which calls the Finvoice XMLport. This runs the XMLport and saves the XML file generated in the temporary folder on the user's computer. The XML Instances closes. If the file is exported, the system returns a success message. Otherwise, it returns an error message.

```
OnRun()
FinvoiceXmlFile.CREATE(TEMPORARYPATH + 'Finvoice.xml');
FinvoiceXmlFile.CREATEOUTSTREAM(XmlOutStream);
Sent := XMLPORT.EXPORT(XMLPORT::FinvoiceEdited, XmlOutStream);
OutputFile := FinvoiceXmlFile.NAME;
InputFile := 'Finvoice.xml';
FinvoiceXmlFile.CLOSE;
IF Sent THEN
BEGIN
  DOWNLOAD(OutputFile,'Download file','C:\Temp','Xml file(*.xml)|*.xml',InputFile);
  ERASE(OutputFile);
  MESSAGE(Text000);
END
ELSE
  MESSAGE(Text001);
```

(MSDN 2016g)

5 Result Evaluation

The implementation of Finvoice in Microsoft Dynamics NAV involves a whole lot of works and the works are broken down into stages. As a result, there are different types of results expected at every stage of the project lifecycle.
The actual result is a valid Finvoice 2.01 XML file but before getting to this stage, it is important to know that the final result is the combination of results from other stages.

When a sales order is posted and a posted invoice is generated, the invoice system is expected to update the Finvoice tables.

The system was tested by creating posting existing sales orders severally and the Finvoice were updated each as expected.

Object Type	Object Name	Updated
Table	MessageTransmissionDetails	Updated Successfully
Table	SellerPartyDetails	Updated Successfully
Table	BuyerPartyDetails	Updated Successfully
Table	AnyPartyDetails	Updated Successfully
Table	InvoiceDetails	Updated Successfully
Table	InvoiceRow	Updated Successfully
Table	InvoiceRow II	Updated Successfully
Table	EpiDetails	

5.1 System Test

The system was tested a few times and below is a visual representation of the result.

A Sales Order was created as shown in the figure below:

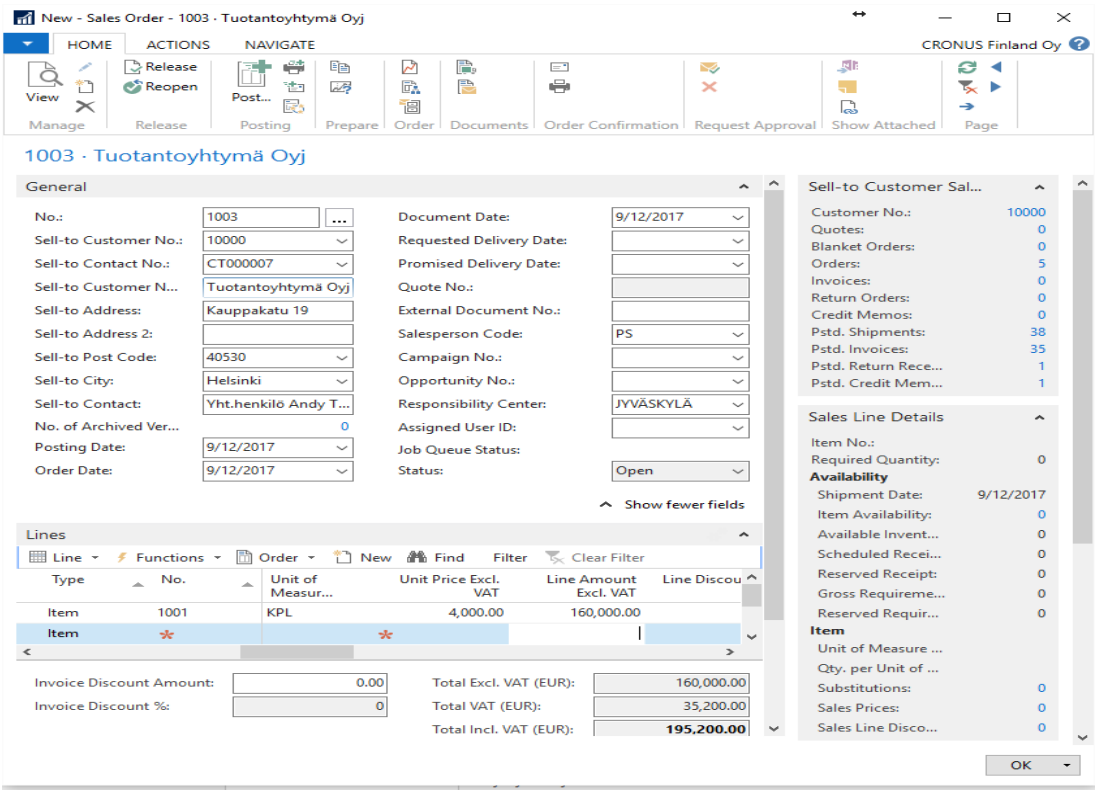

Figure 5-1 Sales Order

Post shipment and post invoice

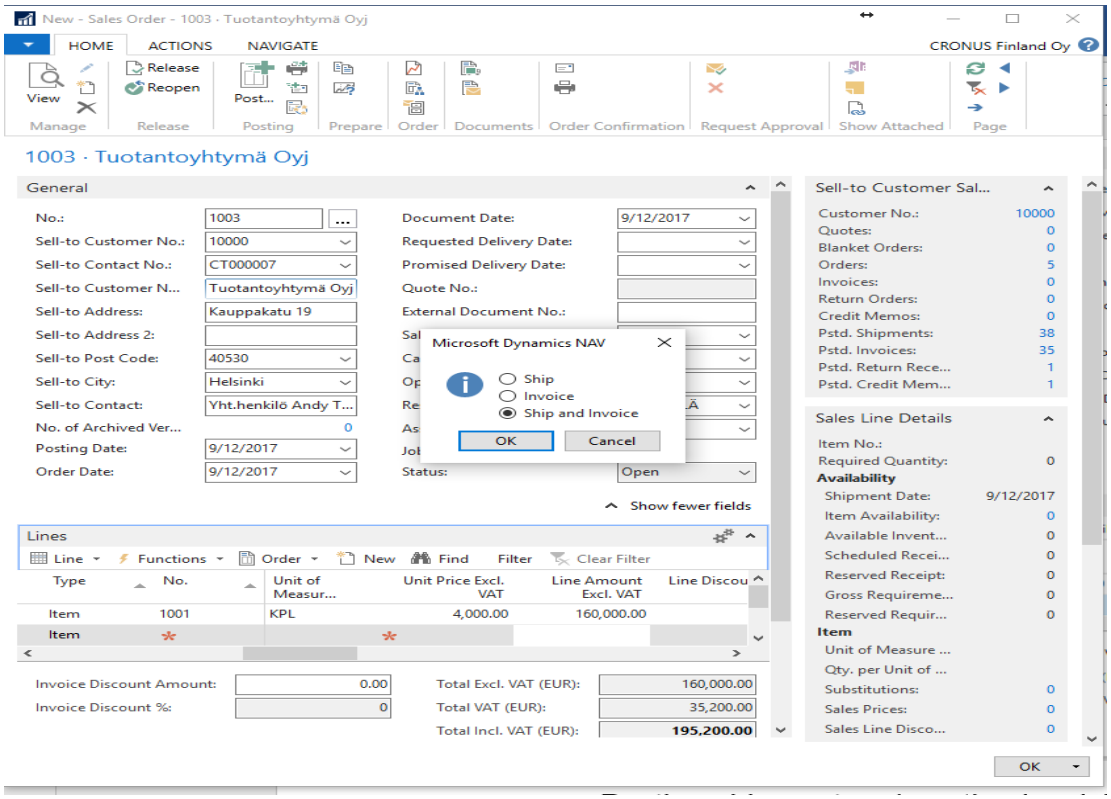

Figure 5-2 Post Sales Order

The invoice was posted and the finvoice XML file is created with a message showing that:

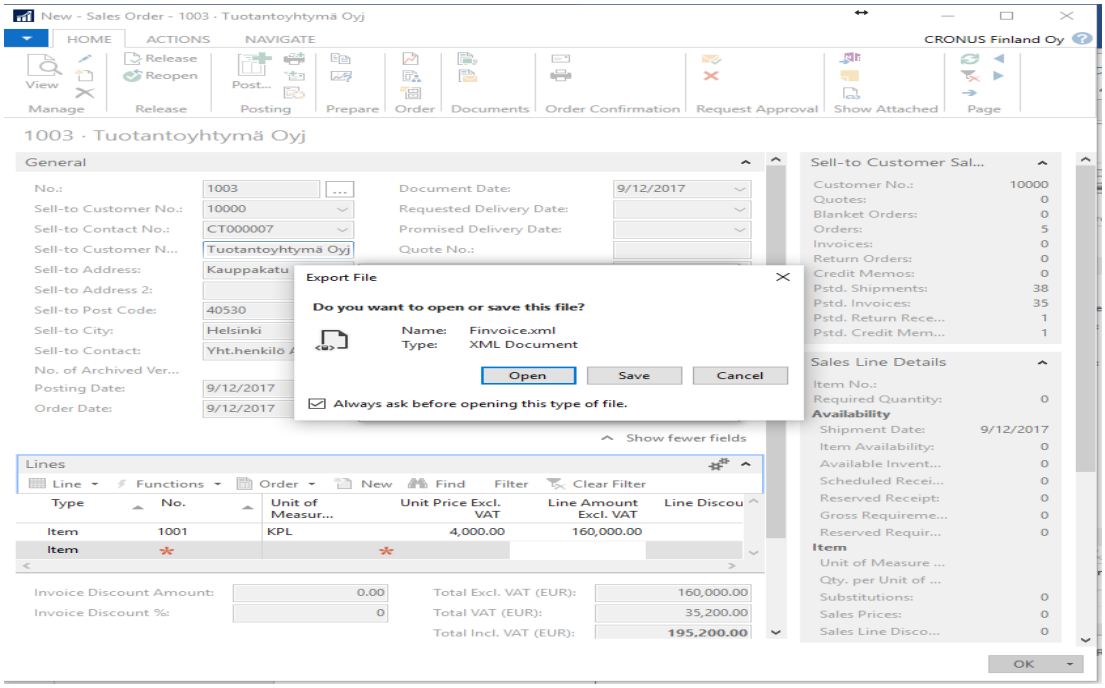

Figure 5-3 Finvoice Creation Notification

Click open to view the XML file created. If the XML file is created successfully, the system sends a message confirming that it was successfully exported.

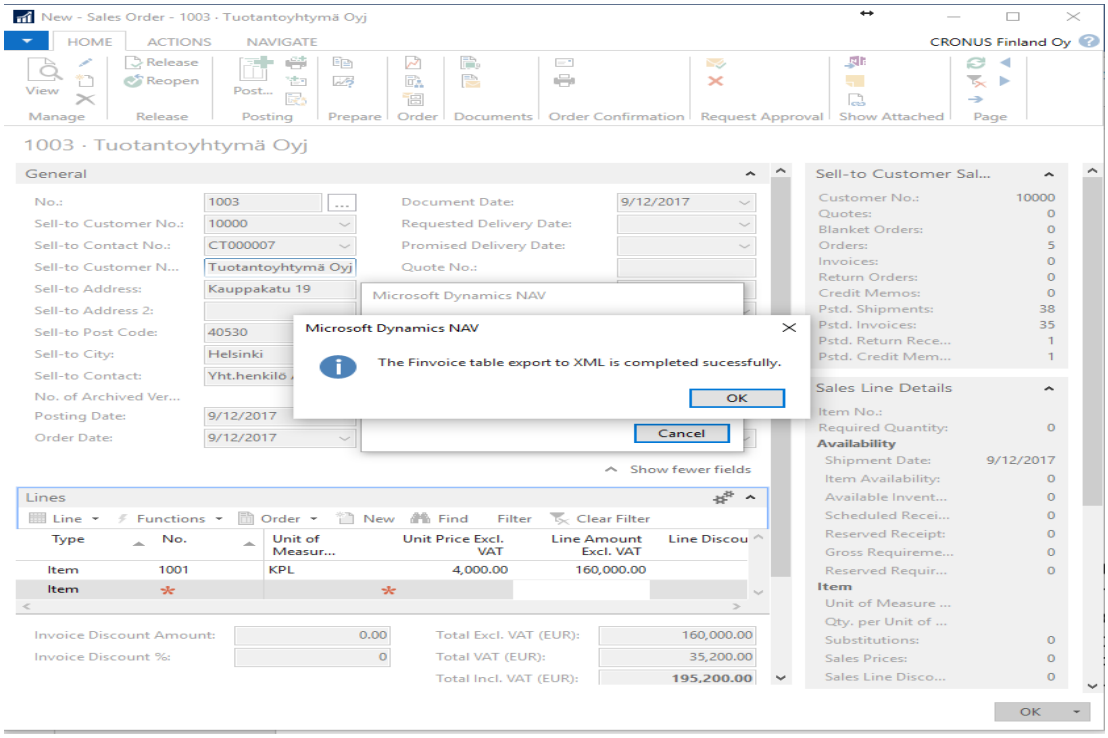

Figure 5-4 Export Confirmation

It will open in Visual Studio because I have set visual studio as the default app to open it.
The figure below shows the Finvoice XML generated from Microsoft Dynamics NAV 2016. Some fields are not updated yet because a decision needs to be made later about where the information will come from in Microsoft Dynamics NAV.

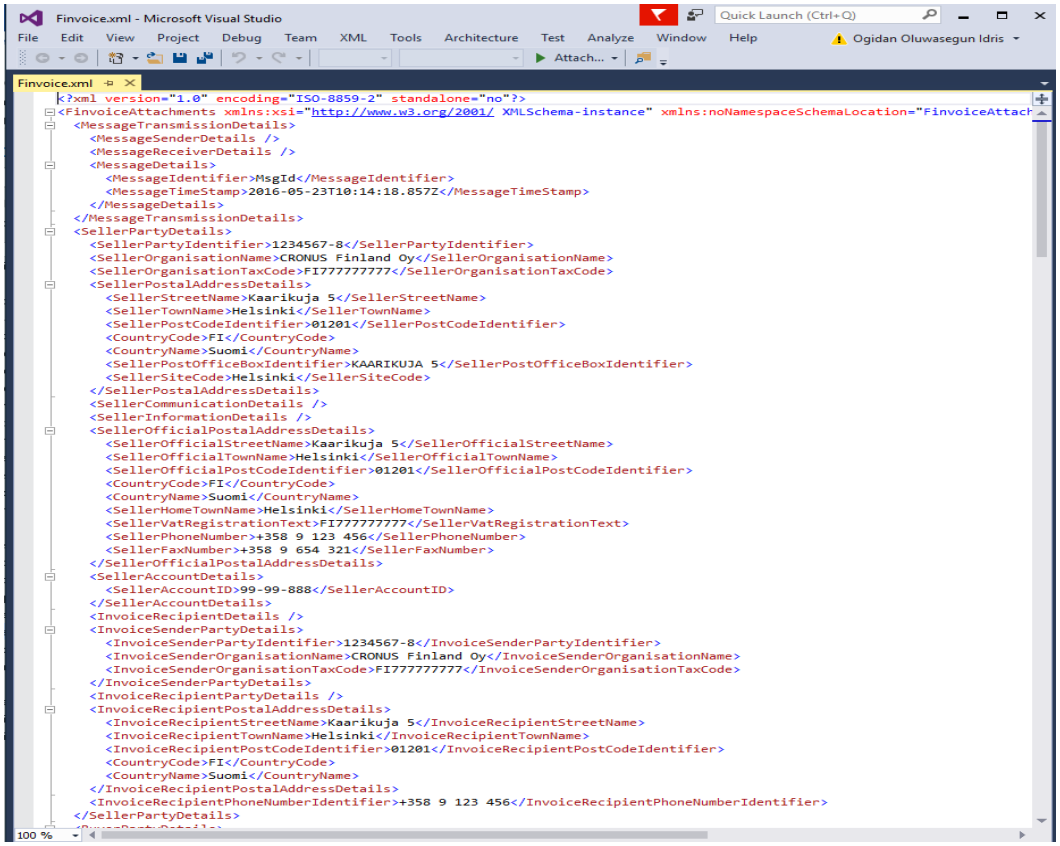

Figure 5-5 Finvoice XML Part I

The figure below contains the invoice details and the values are from Microsoft Dynamics NAV.

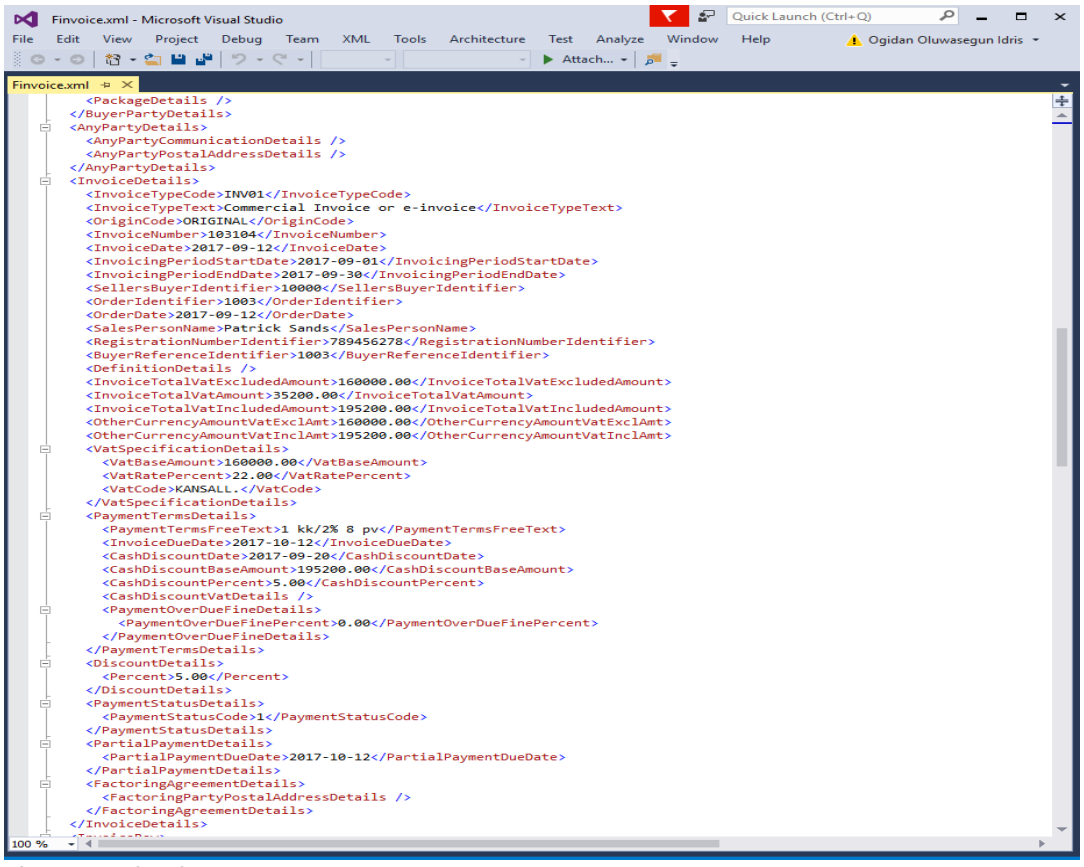

Figure 5-6 Finvoice XML Part 2

5.2 Validating Finvoice

The Finvoice XML file which is the final result of the project is being generated when a sales order is invoiced but the generated XML file needs to be checked and validated. Upon passing the checks and validation, then it is said to be a valid Finvoice 2.01 XML file.
Finvoice XML file can be validated using either Truugo or XML Validation. The final Finvoice will not be *validated* because the XMLport need some customizations so that the invoice generated is validated.

6 Conclusion

Tero Hassinen, a Senior Microsoft Dynamics NAV Developer at Navpartneri Oy came up with the idea of this implementing Finvoice in Microsoft Dynamics NAV. I have prior experience when I was working a Microsoft Dynamics NAV developer with a Microsoft Partner. I also gained some knowledge in school where I was taught some very closely related courses such as ERP Basics, ERP Advanced, Supply Chain Management, Managing Business Information Technology, Business Process Modeling, Business intelligence in which Microsoft Dynamics NAV was used as one of the main working ERP systems.

My prior knowledge and experience gave me the courage to work on this project. It has been a good experience for me considering the fact that I have not been actively involved in Microsoft Dynamics NAV project for the past four years.
The desired goal was to customize the Microsoft Dynamics NAV application such that when a sales order is posted and a sales invoice is generated, a Finvoice XML file is generated.
Right now, the system works in such a way that the Finvoice XML file is generated once a sales invoice is generated but the generated XML file needs to be further checked and validated.

I gained a lot during this project because I learned a lot of new things in the process. I learned more about prioritization during projects and I realized the importance of having a good project plan before implementing a project. It is very good to implement the project with the already written project plan as it will serve as a guide throughout the lifecycle of this project

6.1 Future Research and Development

The project did no go according to initial plan due to some unexpected change of situation which was unavoidable. The situation of Tero Hassinen who came up with the idea of idea of this project changed and a result it was difficult to get things done as I had to do everything almost single-handedly.
The desired final result was only close to being achieved and as a result, there will be a need further research and development.

References

Federation of Finnish Financial Services FFI 2015, Finvoice Implementation Guidelines. URL:http://www.finanssiala.fi/finvoice/dokumentit/Finvoice_2_01_implementation_guidelines.pdf. Accessed: 3 March 2016.

Microsoft. 80439_ NAV2013_ENUS_INTRO. Module 1: Microsoft Dynamics NAV 2013 as an ERP System. Accessed: 15 May 2016.

MSDN 2016a. Microsoft Dynamics NAV 2016. URL: https://msdn.microsoft.com/en-us/library/hh173988(v=nav.90).aspx. Accessed: 7 March 2016.

MSDN 2016b. Microsoft Dynamics NAV Product and Architecture Overview. URL: https://msdn.microsoft.com/en-us/library/dd354965(v=nav.90).aspx. Accessed. 7 April 2016.

MSDN 2016c. How to: Upload the License File. URL: https://msdn.microsoft.com/en-us/library/dd338774(v=nav.90).aspx. Accessed: 15 April 2016.

MSDN 2016d. How to: Install C/SIDE Development Environment. URL: https://msdn.microsoft.com/en-us/library/dd338774(v=nav.90).aspx. Accessed: 14 April 2016.

MSDN 2016e. License Types. URL: https://msdn.microsoft.com/en-us/library/jj551750(v=nav.90).aspx. Accessed: 20 April 2016.

MSDN 2016f. Creating XMLports. URL: https://msdn.microsoft.com/en-us/library/dd338846.aspx. Accessed: 20 January 2016.

MSDN 2016g. How to: Developing Codeunit to Run XMLport. URL: https://msdn.microsoft.com/en-us/library/dd355410.aspx. Accessed: 20 January 2016.

MSDN 2016h. Walkthrough: Installing Demo Version. URL: https://msdn.microsoft.com/en-us/library/dd301212.aspx. Accessed: 18 January 2016

OpenText Corporation 2015. E-invoicing. URL: http://www.gxs.co.uk/eBooks/eInvoicing/e-invoicing_ebook_uk.pdf. Accessed: 5 March 2016.

OpenText Corporation 2016. What is e-invoicing? URL: http://www.einvoicingbasics.co.uk/what-is-e-invoicing/. Accessed: 5 March 2016.

OpenText Corporation 2016. Type of e-invoice. URL: http://www.einvoicingbasics.co.uk/what-is-e-invoicing/types-of-einvoice/. Accessed: 5 March 2016.

Rochelle P. Cohen. EDI Basics. How successful Businesses Connect, Communicate, and Collaborate around the world. Accessed: 14 May 2016.

W3School 2016. Introduction to XML. URL: http://www.w3schools.com/xml/xml_whatis.asp. Accessed: 17 May 2016.

Appendices

Appendix 1. Finvoice Structure

Appendix 2. Finvoice Functions

```
LOCAL InsertMessageTransmissionDetails(CompanyInfo : Record "Company
Information";Cust : Record Customer;VAR MessageTransmissionDetails : Record
MessageTransmissionDetails)
WITH MessageTransmissionDetails DO BEGIN
  MessageTransmissionDetails.INIT;
  MessageTransmissionDetails.InvoiceNumber := SalesInvHeader."No.";
  MessageTransmissionDetails.FromIdentifier := CompanyInfo."Business Identity
Code";
  MessageTransmissionDetails.FromIntermediator := CompanyInfo."SWIFT Code";
  MessageTransmissionDetails.ToIdentifier := Cust."No."; //To be continued!
  CustBankAcct.SETRANGE("Customer No.",Cust."No.");
  IF CustBankAcct.FINDFIRST THEN BEGIN REPEAT
    IF Cust.Name = CustBankAcct.Name THEN
      MessageTransmissionDetails.ToIntermediator :=CustBankAcct."SWIFT Code";
    UNTIL CustBankAcct.NEXT = 0;
  END;
  MessageTransmissionDetails.ImplementationCode := '';
  MessageTransmissionDetails.MessageIdentifier := 'MsgId' +
CompanyInfo."Business Identity Code";
  MessageTransmissionDetails.RefToMessageIdentifier := ''; //To be reviewed later
  MessageTransmissionDetails.MessageTimeStamp := CURRENTDATETIME;
  MessageTransmissionDetails.INSERT;
END;

LOCAL InsertSellerPartyDetails(CompanyInfo : Record "Company
Information";VAR SellerPartyDetails : Record SellerPartyDetails;VAR Country :
Record "Country/Region")
CompanyInfo.GET();
IF NOT SellerPartyDetails.GET(CompanyInfo."Business Identity Code") THEN
BEGIN
  SellerPartyDetails.INIT;
  SellerPartyDetails.SellerPartyIdentifier := CompanyInfo."Business Identity Code";
  SellerPartyDetails.SellerPartyIdentifierUrlText := '';
  SellerPartyDetails.SellerOrganisationName := CompanyInfo.Name;
  SellerPartyDetails.SellerOrganisationTaxCode := CompanyInfo."VAT Registration
No.";
  SellerPartyDetails.SellerOrganisationTaxCodeUrlTe := CompanyInfo."Home
Page";
  SellerPartyDetails.SellerCode := CompanyInfo.GLN;
  SellerPartyDetails.SellerStreetName := CompanyInfo.Address;
  SellerPartyDetails.SellerTownName := CompanyInfo."Address 2";
  SellerPartyDetails.VALIDATE(SellerPostCodeIdentifier, CompanyInfo."Post
Code");
  SellerPartyDetails.VALIDATE(CountryCode, CompanyInfo."Country/Region
Code");
  IF Country.GET(CompanyInfo."Country/Region Code") THEN
    SellerPartyDetails.CountryName := Country.Name;
  SellerPartyDetails.SellerPostOfficeBoxIdentifier := CompanyInfo.Address;
  SellerPartyDetails.SellerOrganisationUnitNumber := '';
  SellerPartyDetails.SellerSiteCode := CompanyInfo.City;
  SellerPartyDetails.SellerContactPersonName := Cust.Contact;
  IF Contact.GET(Cust."Primary Contact No.") THEN BEGIN
    SellerPartyDetails.SellerContactPersonFunction := Contact."Job Title";
    SellerPartyDetails.SellerContactPersonDepartment := '';
```

```
  END;
  SellerPartyDetails.SellerPhoneNumberIdentifier := CompanyInfo."Phone No. 2";
  SellerPartyDetails.SellerEmailaddressIdentifier := CompanyInfo."E-Mail";
  SellerPartyDetails.SellerOfficialStreetName := CompanyInfo.Address;
  SellerPartyDetails.SellerOfficialTownName := CompanyInfo.City;
  SellerPartyDetails.SellerOfficialPostCodeIdentifi := CompanyInfo."Post Code";
  SellerPartyDetails.CountryCode_SellerInfoDetails :=
CompanyInfo."Country/Region Code";//Change of name might be necessary
  SellerPartyDetails.CountryName_SellerInfoDetails := Country.Name;
  SellerPartyDetails.SellerHomeTownName := CompanyInfo.City;
  SellerPartyDetails.SellerVatRegistrationText := CompanyInfo."VAT Registration
No.";
  SellerPartyDetails.SellerVatRegistrationDate := CompanyInfo."Customs Permit
Date";//Change of name might be necessary
  SellerPartyDetails.SellerTaxRegistrationText := ''; //Not used in finland (Swedish
Tax Demand Note)
  SellerPartyDetails.SellerPhoneNumber := CompanyInfo."Phone No.";
  SellerPartyDetails.SellerFaxNumber := CompanyInfo."Fax No.";
  SellerPartyDetails.SellerCommonEmailaddressIdenti := CompanyInfo."E-Mail";
  SellerPartyDetails.SellerWebaddressIdentifier := CompanyInfo."Home Page";
  SellerPartyDetails.SellerFreeText := CompanyInfo."Name 2";//Change of name
might be necessary
  SellerPartyDetails.SellerAccountID := CompanyInfo."Bank Account No.";
  SellerPartyDetails.SellerBic := CompanyInfo."SWIFT Code";
  SellerPartyDetails.InvoiceRecipientAddress := CompanyInfo."E-Mail"; //Change
of name might be necessary
  SellerPartyDetails.InvoiceRecipientIntermediatAdd := '';
  SellerPartyDetails.InvoiceSenderPartyIdentifier := CompanyInfo."Business
Identity Code";
  SellerPartyDetails.InvoiceSenderOrganisationName := CompanyInfo.Name; //To
be reviewed later.
  SellerPartyDetails.InvoiceSenderOrganisationTaxCo := CompanyInfo."VAT
Registration No.";
  SellerPartyDetails.InvoiceSenderCode := CompanyInfo.GLN;
  SellerPartyDetails.InvoiceRecipientStreetName := CompanyInfo.Address;
  SellerPartyDetails.InvoiceRecipientTownName := CompanyInfo.City;
  SellerPartyDetails.InvoiceRecipientPostCodeIdenti := CompanyInfo."Post Code";
  SellerPartyDetails.CountryCode_InvRecPAddrDetails :=
CompanyInfo."Country/Region Code";
  IF Country.GET(CompanyInfo."Country/Region Code") THEN
    SellerPartyDetails.CountryName_InvRecPAddrDetails := Country.Name;
  SellerPartyDetails.InvoiceRecipientPostOffBoxIden := '';
  SellerPartyDetails.InvoiceRecipientOrganUnitNo := CompanyInfo."Company Reg.
No."; //To be reviewed later.
  SellerPartyDetails.InvoiceRecipientSiteCode := CompanyInfo."Location Code";
  SellerPartyDetails.InvoiceRecipientContactPersonN := CompanyInfo."Ship-to
Contact";
  SellerPartyDetails.InvoiceRecipientContactFunctn := '';
  SellerPartyDetails.InvoiceRecipientContactDepartm := ''; //CheckLater
  SellerPartyDetails.InvoiceRecipientLanguageCode := ''; //checker
  SellerPartyDetails.InvoiceRecipientPhoneNumberIde := CompanyInfo."Phone
No.";
  SellerPartyDetails.InvoiceRecipientEmailaddressId := CompanyInfo."E-Mail";
  SellerPartyDetails.INSERT;
END;
```

```
LOCAL InsertBuyerPartyDetails(Cust : Record Customer;VAR BuyerPartyDetails :
Record BuyerPartyDetails)

IF NOT BuyerPartyDetails.GET(SalesInvHeader."Sell-to Customer No.") THEN
BEGIN
  Cust.GET(SalesInvHeader."Sell-to Customer No.");
  BuyerPartyDetails.INIT;
  BuyerPartyDetails.BuyerPartyIdentifier := Cust."No.";
  BuyerPartyDetails.BuyerOrganisationName := Cust.Name;
  BuyerPartyDetails.BuyerOrganisationDepartment := Cust."Global Dimension 1
Code";
  BuyerPartyDetails.BuyerOrganisationTaxCode := Cust."VAT Registration No.";
  BuyerPartyDetails.BuyerCode := Cust.GLN;
  BuyerPartyDetails.BuyerStreetName := Cust.Address;
  BuyerPartyDetails.BuyerTownName := Cust.City;
  BuyerPartyDetails.BuyerPostCodeIdentifier := Cust."Post Code";
  BuyerPartyDetails.CountryCode_BuyerPostalAddrDet := Cust."Country/Region
Code";
  IF Country.GET(Cust."Country/Region Code") THEN
    BuyerPartyDetails.CountryName_BuyerPostalAddrDet := Country.Name;
  BuyerPartyDetails.BuyerPostOfficeBoxIdentifier := '';
  BuyerPartyDetails.BuyerOrganisationUnitNumber := '';
  BuyerPartyDetails.BuyerSiteCode := Cust."Location Code";
  BuyerPartyDetails.BuyerContactPersonName := Cust.Contact;
  BuyerPartyDetails.BuyerContactPersonFunction := '';
  BuyerPartyDetails.BuyerContactPersonDepartment := '';
  BuyerPartyDetails.BuyerPhoneNumberIdentifier := Cust."Phone No.";
  BuyerPartyDetails.BuyerEmailaddressIdentifier := Cust."E-Mail";
  BuyerPartyDetails.DeliveryPartyIdentifier := '';
  BuyerPartyDetails.DeliveryOrganisationName := '';
  BuyerPartyDetails.DeliveryOrganisationDepartment := '';
  BuyerPartyDetails.DeliveryOrganisationTaxCode := '';
  BuyerPartyDetails.DeliveryCode := '';
  BuyerPartyDetails.DeliveryStreetName := '';
  BuyerPartyDetails.DeliveryTownName := '';
  BuyerPartyDetails.DeliveryPostCodeIdentifier := '';
  BuyerPartyDetails.CountryCode_DeliveryPartyDet := '';
  BuyerPartyDetails.CountryName_DeliveryPartyDet := '';
  BuyerPartyDetails.DeliveryPostofficeBoxIdentifie := '';
  BuyerPartyDetails.DeliveryOrganisationUnitNumber := '';
  BuyerPartyDetails.DeliverySiteCode := '';
  BuyerPartyDetails.DeliveryContactPersonName := '';
  BuyerPartyDetails.DeliveryContactPersonFunction := '';
  BuyerPartyDetails.DeliveryContactPersonDepartmen := '';
  BuyerPartyDetails.DeliveryPhoneNumberIdentifier := '';
  BuyerPartyDetails.DeliveryEmailaddressIdentifier := '';
  BuyerPartyDetails.DeliveryDate := 0D;
  BuyerPartyDetails.StartDate := 0D;
  BuyerPartyDetails.EndDate := 0D;
  BuyerPartyDetails.ShipmentCode := ''; //To be reviewed later.
  BuyerPartyDetails.ShipmentStreetName := '';
  BuyerPartyDetails.ShipmentTownName := '';
  BuyerPartyDetails.ShipmentPostCodeIdentifier := SalesShptHeader."Shipping
Agent Code";
  IF ShippingAgent.GET(SalesShptHeader."Shipping Agent Code") THEN
    BuyerPartyDetails.ShipmentOrganisationName := ShippingAgent.Name;
```

```
  BuyerPartyDetails.ShipmentOrganisationDepartment := '';
  BuyerPartyDetails.ShipmentOrganisationTaxCode := '';
  BuyerPartyDetails.ShipmentCode := ''; //To be reviewed
  BuyerPartyDetails.ShipmentStreetName := '';
  BuyerPartyDetails.ShipmentTownName := '';
  BuyerPartyDetails.ShipmentPostCodeIdentifier := '';
  BuyerPartyDetails.CountryCode_ShipmentPartyDet := '';
  BuyerPartyDetails.CountryName_ShipmentPartyDet := '';
  BuyerPartyDetails.ShipmentPostOfficeBoxIdentifie := '';
  BuyerPartyDetails.ShipmentSiteCode := '';
  IF ShipmentMethod.GET(SalesShptHeader."Shipment Method Code") THEN
    BuyerPartyDetails.DeliveryMethodText := ShipmentMethod.Description;
  BuyerPartyDetails.DeliveryTermsCode := '';
  BuyerPartyDetails.DeliveryTermsText := '';
  BuyerPartyDetails.DeliveryTermsCode := '';
  BuyerPartyDetails.TerminalAddressText := '';
  BuyerPartyDetails.WaybillIdentifier := '';
  BuyerPartyDetails.WaybillTypeCode := '';
  BuyerPartyDetails.ClearanceIdentifier := '';
  BuyerPartyDetails.DeliveryNoteIdentifier := '';
  BuyerPartyDetails.DelivererIdentifier := '';
  BuyerPartyDetails.DelivererName := '';
  BuyerPartyDetails.DelivererCountryCode := '';
  BuyerPartyDetails.DelivererCountryName := '';
  BuyerPartyDetails.ModeOfTransportIdentifier := '';
  BuyerPartyDetails.CarrierName := '';
  BuyerPartyDetails.VesselName := '';
  BuyerPartyDetails.LocationIdentifier := '';
  BuyerPartyDetails.TransportInformationDate := 0D;
  BuyerPartyDetails.CountryOfOrigin := '';
  BuyerPartyDetails.CountryOfDestinationName := '';
  BuyerPartyDetails.DestinationCountryCode := '';
  BuyerPartyDetails.PlaceOfDischarge := '';
  BuyerPartyDetails.FinalDestinationName := '';
  BuyerPartyDetails.ManufacturerIdentifier := '';
  BuyerPartyDetails.ManufacturerName := '';
  BuyerPartyDetails.ManufacturerCountryCode := '';
  BuyerPartyDetails.ManufacturerCountryName := '';
  BuyerPartyDetails.ManufacturerOrderIdentifier := '';
  BuyerPartyDetails.PackageLength := 0;
  BuyerPartyDetails.PackageWidth := 0;
  BuyerPartyDetails.PackageHeight := 0;
  BuyerPartyDetails.PackageWeight := 0;
  BuyerPartyDetails.PackageNetWeight := 0;
  BuyerPartyDetails.PackageVolume := 0;
  BuyerPartyDetails.TransportCarriageQuantity := 0;
  BuyerPartyDetails.INSERT;
END;

LOCAL InsertAnyPartyDetails(VAR AnyPartyDetails : Record AnyPartyDetails)
IF NOT AnyPartyDetails.GET(SalesInvHeader."Sell-to Customer No.") THEN
BEGIN
  AnyPartyDetails.INIT;
  AnyPartyDetails.BuyerPartyIdentifier := SalesInvHeader."Sell-to Customer No.";
  AnyPartyDetails.AnyPartyText := '';
  AnyPartyDetails.AnyPartyIdentifier := '';
```

```
  AnyPartyDetails.AnyPartyOrganisationName := '';
  AnyPartyDetails.AnyPartyOrganisationDepartment := '';
  AnyPartyDetails.AnyPartyOrganisationTaxCode := '';
  AnyPartyDetails.AnyPartyCode := '';
  AnyPartyDetails.AnyPartyContactPersonName := '';
  AnyPartyDetails.AnyPartyContactPersonFunction := '';
  AnyPartyDetails.AnyPartyContactPersonDepartmen := '';
  AnyPartyDetails.AnyPartyPhoneNumberIdentifier := '';
  AnyPartyDetails.AnyPartyEmailAddressIdentifier := '';
  AnyPartyDetails.AnyPartyStreetName := '';
  AnyPartyDetails.AnyPartyTownName := '';
  AnyPartyDetails.AnyPartyPostCodeIdentifier := '';
  AnyPartyDetails.AnyPartyCountryCode := '';
  AnyPartyDetails.AnyPartyCountryName := '';
  AnyPartyDetails.AnyPartyPostOfficeBoxIdentifie := '';
  AnyPartyDetails.AnyPartyOrganisationUnitNumber := '';
  AnyPartyDetails.AnyPartySiteCode := '';
  AnyPartyDetails.INSERT;
END;

LOCAL InsertInvoiceDetails(SalesInvoiceHeader : Record "Sales Invoice
Header";VAR InvoiceDetails : Record InvoiceDetails)
InvoiceDetails.INIT;
InvoiceDetails.VALIDATE(InvoiceTypeCode,'INV01');
IF InvoiceTypeCode.GET(InvoiceDetails.InvoiceTypeCode) THEN
  InvoiceDetails.InvoiceTypeText := InvoiceTypeCode.Description;
InvoiceDetails.VALIDATE(OriginCode,'Original');
InvoiceDetails.InvoiceNumber := SalesInvHeader."No.";
InvoiceDetails.InvoiceDate := SalesInvHeader."Posting Date";
InvoiceDetails.OriginalInvoiceNumber := ''; //To be reviewed later.
MonthofDate := DATE2DMY(SalesInvHeader."Posting Date",2);
InvoiceDetails.InvoicingPeriodStartDate := CALCDATE('<-
CM>',SalesInvHeader."Posting Date");
InvoiceDetails.InvoicingPeriodEndDate :=
CALCDATE('<CM>',SalesInvHeader."Posting Date");
InvoiceDetails.SellerReferenceIdentifier := SalesInvHeader."External Document
No.";
InvoiceDetails.SellerReferenceIdentifierUrlTe := CompanyInfo."Home Page";
InvoiceDetails.BuyersSellerIdentifier := ''; //To be reviewed later
InvoiceDetails.SellersBuyerIdentifier := SalesInvHeader."Sell-to Customer No.";
InvoiceDetails.OrderIdentifier := SalesInvHeader."Order No.";
InvoiceDetails.OrderIdentifierUrlText := ''; //To be reviewed later
InvoiceDetails.OrderDate := SalesInvHeader."Order Date";
IF SalesPerson.GET(SalesInvHeader."Salesperson Code") THEN
  InvoiceDetails.SalesPersonName := SalesPerson.Name;
InvoiceDetails.OrderConfirmationIdentifier := '';
InvoiceDetails.OrderConfirmationDate := 0D;
InvoiceDetails.AgreementIdentifier := '';
InvoiceDetails.AgreementIdentifierUrlText := '';
InvoiceDetails.AgreementTypeText := '';
InvoiceDetails.AgreementTypeCode := '';
InvoiceDetails.AgreementDate := 0D;
InvoiceDetails.NotificationIdentifier := '';
InvoiceDetails.NotificationDate := 0D;
InvoiceDetails.RegistrationNumberIdentifier := SalesInvHeader."VAT Registration
No.";
```

```
InvoiceDetails.ControllerIdentifier := '';
InvoiceDetails.ControllerName := '';
InvoiceDetails.ControlDate := 0D;
InvoiceDetails.BuyerReferenceIdentifier := SalesHeader."No.";
InvoiceDetails.ProjectReferenceIdentifier := '';
InvoiceDetails.DefinitionHeaderText := '';
InvoiceDetails.DefinitionValue := '';
SalesInvoiceLine.SETRANGE(SalesInvoiceLine."Document No.",
SalesInvHeader."No.");
IF SalesInvoiceLine.FINDFIRST THEN BEGIN
  REPEAT
    TotalInvoiceAmountIncludingVAT := TotalInvoiceAmountIncludingVAT +
SalesInvoiceLine."Amount Including VAT";
    TotalInvoiceAmountExcludingVAT := TotalInvoiceAmountExcludingVAT +
SalesInvoiceLine.Amount;
  UNTIL SalesInvoiceLine.NEXT = 0;
END;
InvoiceDetails.InvoiceTotalVatExcludedAmount :=
TotalInvoiceAmountExcludingVAT;
InvoiceDetails.InvoiceTotalVatIncludedAmount :=
TotalInvoiceAmountIncludingVAT;
InvoiceDetails.InvoiceTotalVatAmount := TotalInvoiceAmountIncludingVAT -
TotalInvoiceAmountExcludingVAT;
InvoiceDetails.InvoiceTotalRoundoffAmount := 0; //To be reviewed later
CurrencyExchangeRate.SETRANGE(CurrencyExchangeRate."Currency Code",
SalesInvHeader."Currency Code");
CurrencyExchangeRate.SETFILTER(CurrencyExchangeRate."Starting
Date",'<%1', SalesInvHeader."Posting Date");
IF CurrencyExchangeRate.FINDLAST THEN
  InvoiceDetails.ExchangeRate := CurrencyExchangeRate."Exchange Rate
Amount";
IF SalesInvHeader."Currency Code" = '' THEN BEGIN
  InvoiceDetails.OtherCurrencyAmountVatExclAmt :=
TotalInvoiceAmountExcludingVAT;
  InvoiceDetails.OtherCurrencyAmountVatInclAmt :=
TotalInvoiceAmountIncludingVAT;
END ELSE BEGIN
  InvoiceDetails.OtherCurrencyAmountVatExclAmt :=
TotalInvoiceAmountExcludingVAT / InvoiceDetails.ExchangeRate; //In LCY
  InvoiceDetails.OtherCurrencyAmountVatInclAmt :=
TotalInvoiceAmountIncludingVAT / InvoiceDetails.ExchangeRate; //In LCY
END;
InvoiceDetails.CreditLimitAmount := Cust."Credit Limit (LCY)";
InvoiceDetails.CreditInterestPercent := 0;
InvoiceDetails.OperationLimitAmount := 0; //TBR Later
InvoiceDetails.MonthlyAmount := 0;
InvoiceDetails.ShortProposedAccountIdentifier := SalesInvHeader."Bal. Account
No.";
InvoiceDetails.NormalProposedAccountIdentifie := SalesInvHeader."Bal. Account
No.";
InvoiceDetails.ProposedAccountText := '';
InvoiceDetails.AccountDimensionText := '';
InvoiceDetails.SellerAccountText := '';
InvoiceDetails.VatBaseAmount := TotalInvoiceAmountExcludingVAT;
```

```
InvoiceDetails.VatRatePercent :=
(InvoiceDetails.InvoiceTotalVatAmount/InvoiceDetails.InvoiceTotalVatExcludedAm
ount) * 100;
InvoiceDetails.VatCode := SalesInvHeader."VAT Bus. Posting Group";
InvoiceDetails.VatFreeText := '';//Must Check
InvoiceDetails.InvoiceVatFreeText := ''; //Must Check
IF PaymentTerms.GET(SalesInvHeader."Payment Terms Code") THEN
  InvoiceDetails.PaymentTermsFreeText := PaymentTerms.Description;
InvoiceDetails.InvoiceDueDate := SalesInvHeader."Due Date";
InvoiceDetails.CashDiscountDate := SalesInvHeader."Pmt. Discount Date";
InvoiceDetails.CashDiscountBaseAmount :=
InvoiceDetails.InvoiceTotalVatIncludedAmount;
CustInvoiceDisc.SETRANGE(CustInvoiceDisc.Code,SalesInvHeader."Invoice
Disc. Code");
CustInvoiceDisc.SETRANGE(CustInvoiceDisc."Currency
Code",SalesInvHeader."Currency Code");
IF CustInvoiceDisc.FINDFIRST THEN
  InvoiceDetails.CashDiscountPercent := CustInvoiceDisc."Discount %";
InvoiceDetails.CashDiscountAmount := SalesInvHeader."Invoice Discount
Amount";
InvoiceDetails.CashDiscountExcludingVatAmount := 0; //Must Check
InvoiceDetails.CashDiscountVatPercent := 0; //Must Check
InvoiceDetails.CashDiscountVatAmount := 0; //Must Check
InvoiceDetails.ReducedInvoiceVatIncludedAmt := 0; //Must Check
InvoiceDetails.PaymentOverDueFineFreeText := ''; //Must Check
InvoiceDetails.PaymentOverDueFinePercent := 0; //Must Check
InvoiceDetails.PaymentOverDueFixedAmount := 0; //Must Check
InvoiceDetails.FreeText := '' + FORMAT(InvoiceDetails.CashDiscountPercent);
InvoiceDetails.Percent := CustInvoiceDisc."Discount %"; //Confusing with
CashDiscountPercent
InvoiceDetails.Amount := SalesInvHeader."Invoice Discount Amount";
IF SalesInvHeader."Remaining Amount" = 0 THEN
  InvoiceDetails.PaymentStatusCode := 1
ELSE IF SalesInvHeader."Remaining Amount" = SalesInvHeader."Amount
Including VAT" THEN
   InvoiceDetails.PaymentStatusCode := 2
ELSE IF SalesInvHeader."Remaining Amount" < SalesInvHeader."Amount
Including VAT" THEN
   InvoiceDetails.PaymentStatusCode := 3;
IF InvoiceDetails.GET(SalesInvHeader."Payment Method Code") THEN
  InvoiceDetails.PaymentMethodText := PaymentMethod.Description;
InvoiceDetails.PaidAmount := SalesInvHeader."Amount Including VAT"-
SalesInvHeader."Remaining Amount";
//InvoiceVatPercentage := ((SalesInvHeader."Amount Including VAT"-
SalesInvHeader.Amount)/SalesInvHeader.Amount) * 100;
//InvoiceDetails.PaidVatExcludedAmount := SalesInvHeader.Amount -
((SalesInvHeader."Amount Including VAT" - SalesInvHeader."Remaining Amount")
* InvoiceVatPercentage); // TBR
InvoiceDetails.UnPaidAmount := SalesInvHeader."Remaining Amount";
InvoiceDetails.UnPaidVatExcludedAmount := SalesInvHeader.Amount -
(SalesInvHeader."Amount Including VAT" - SalesInvHeader."Remaining Amount");
//TBR
InvoiceDetails.InterestPercent := 0;
InvoiceDetails.ProcessingCostsAmount := 0;
InvoiceDetails.PartialPaymentVatIncludedAmt := 0;
InvoiceDetails.PartialPaymentVatExcludedAmt := 0;
```

```
InvoiceDetails.PartialPaymentDueDate := SalesInvHeader."Due Date";
InvoiceDetails.PartialPaymentReferenceIdentif := SalesInvHeader."Your
Reference"; //TBR
InvoiceDetails.FactoringAgreementIdentifier := '';
InvoiceDetails.TranmissionListIdentifier := '';
InvoiceDetails.EndorsementClauseCode := '';
InvoiceDetails.FactoringTypeClause := '';
InvoiceDetails.FactoringFreeText := '';
InvoiceDetails.FactoringPartyIdentifier := '';
InvoiceDetails.FactoringPartyName := '';
InvoiceDetails.FactoringPartyStreetName := '';
InvoiceDetails.FactoringPartyTownName := '';
InvoiceDetails.FactoringPartyPostCodeIdentifi := '';
InvoiceDetails.FactoringCountryCode := '';
InvoiceDetails.FactoringCountryName := '';
InvoiceDetails.FactoringPartyPostOfficeBoxIde := '';
InvoiceDetails.VirtualBankBarCode := ''; //Must Check
InvoiceDetails.INSERT;

LOCAL InsertInvoiceRow(SalesInvoiceLine : Record "Sales Invoice Line";VAR
InvoiceRow : Record InvoiceRow;VAR InvoiceRowII : Record "InvoiceRow II")
SalesInvoiceLine.SETRANGE("Document No.",SalesInvHeader."No.");
IF SalesInvoiceLine.FINDFIRST THEN REPEAT
  InvoiceRow.INIT;
  InvoiceRow.RowSubIdentifier := '';
  InvoiceRow.ArticleIdentifier := SalesInvoiceLine."No.";
  InvoiceRow.ArticleGroupIdentifier := SalesInvoiceLine."Gen. Prod. Posting
Group";
  InvoiceRow.ArticleName := SalesInvoiceLine.Description;
  InvoiceRow.ArticleInfoUrlText := ''; //TBR Later
  Item.GET(SalesInvoiceLine."No.");
  InvoiceRow.BuyerArticleIdentifier := Item.GTIN;
  InvoiceRow.EanCode := Item.GTIN;
  InvoiceRow.RowRegistrationNumberIdentifie := ''; //TBR
  InvoiceRow.SerialNumberIdentifier := FORMAT(SalesInvoiceLine."Line No.");
  InvoiceRow.RowActionCode := ''; //TBR
  InvoiceRow.RowDefinitionHeaderText := '';
  InvoiceRow.RowDefinitionValue := '';
  InvoiceRow.OfferedQuantity := SalesInvoiceLine.Quantity;
  InvoiceRow.DeliveredQuantity := SalesInvoiceLine.Quantity;
  InvoiceRow.OrderedQuantity := SalesInvoiceLine.Quantity;
  InvoiceRow.ConfirmedQuantity := SalesInvoiceLine.Quantity;
  //InvoiceRow.CreditRequestedQuantity := 0;
  //InvoiceRow.ReturnedQuantity := 0;
  InvoiceRow.StartDate := CALCDATE('<-CM>',SalesInvoiceLine."Posting Date");
  InvoiceRow.EndDate := CALCDATE('<CM>',SalesInvoiceLine."Posting Date");
  InvoiceRow.UnitPriceAmount := SalesInvoiceLine."Unit Price" -
(SalesInvoiceLine."Unit Price" * SalesInvoiceLine."VAT %");
  InvoiceRow.UnitPriceVatIncludedAmount := SalesInvoiceLine."Unit Price";
  InvoiceRow.UnitPriceBaseQuantity := SalesInvoiceLine."Quantity (Base)";
  InvoiceRow.RowIdentifier := SalesInvoiceLine."Document No.";
  InvoiceRow.RowIdentifierUrlText := '';
  //InvoiceRow.RowOrderPositionIdentifier := FORMAT(SalesInvoiceLine."Line
No."); //TBR
  InvoiceRow.RowIdentifierDate := SalesInvoiceLine."Posting Date";
  InvoiceRow.RowPositionIdentifier := FORMAT(SalesInvoiceLine."Line No.");
```

```
  InvoiceRow.OriginalInvoiceNumber := SalesInvoiceLine."Document No.";
  InvoiceRow.RowOrdererName := SalesHeader."Assigned User ID";
  InvoiceRow.RowSalesPersonName := '';
  InvoiceRow.RowOrderConfirmationIdentifier := SalesInvHeader."Order No.";
  InvoiceRow.RowOrderConfirmationDate := SalesInvHeader."Order Date";
  InvoiceRow.RowDeliveryIdentifier := SalesInvoiceLine."Shipment No.";
  InvoiceRow.RowDeliveryDate := SalesInvoiceLine."Shipment Date";
  InvoiceRow.RowQuotationIdentifier := SalesInvHeader."Quote No.";
  InvoiceRow.RowQuotationIdentifierUrlText := '';
  InvoiceRow.RowAgreementIdentifier := '';
  InvoiceRow.RowAgreementIdentifierUrlText := '';
  InvoiceRow.RowRequestOfQuotationIdentifie := '';
  InvoiceRow.RowRequestOfQuotationIdUrlText := '';
  InvoiceRow.RowPriceListIdentifier := '';
  InvoiceRow.RowPriceListIdentifierUrlText := '';
  InvoiceRow.RowProjectReferenceIdentifier := SalesInvoiceLine."Job No.";
  InvoiceRow.RowOriginalInvoiceIdentifier := SalesInvoiceLine."Document No.";
  InvoiceRow.RowOriginalInvoiceDate := SalesInvHeader."Document Date";
  InvoiceRow.RowOriginalDueDate := SalesInvHeader."Due Date";
  InvoiceRow.RowOriginalInvoiceTotalAmount := SalesInvoiceLine."Amount
Including VAT";
  InvoiceRow.RowOriginalEpiRemittanceInfoId := '';
  InvoiceRow.RowPaidVatExcludedAmount := SalesInvHeader.Amount -
(SalesInvHeader.Amount - SalesInvHeader."Remaining Amount");//TBR
  InvoiceRow.RowPaidVatIncludedAmount := SalesInvHeader."Amount Including
VAT" - (SalesInvHeader."Amount Including VAT" - SalesInvHeader."Remaining
Amount"); //TBR
  InvoiceRow.RowPaidDate := 0D; //TBR
  InvoiceRow.RowUnPaidVatExcludedAmount := 0;//TBR
  InvoiceRow.RowUnPaidVatIncludedAmount := 0;//TBR
  InvoiceRow.RowCollectionDate := SalesInvoiceLine."Shipment Date";
  InvoiceRow.RowCollectionQuantity := SalesInvoiceLine.Quantity; //Shipment Qty
= Invoice Qty
  InvoiceRow.RowCollectionChargeAmount := 0; //TBR
  InvoiceRow.RowInterestRate := 0;
  InvoiceRow.RowInterestStartDate := 0D;
  InvoiceRow.RowInterestEndDate := 0D;
  InvoiceRow.RowInterestPeriodText := '';
  InvoiceRow.RowInterestDateNumber := 0;
  InvoiceRow.RowInterestChargeAmount := 0;
  InvoiceRow.RowInterestChargeVatAmount := 0;
  InvoiceRow.RowAnyPartyText := '';
  InvoiceRow.RowAnyPartyIdentifier := '';
  InvoiceRow.RowAnyPartyOrganisationName := '';
  InvoiceRow.RowAnyPartyOrganisationDepartm := '';
  InvoiceRow.RowAnyPartyOrganisationTaxCode := '';
  InvoiceRow.RowAnyPartyStreetName := '';
  InvoiceRow.RowAnyPartyTownName := '';
  InvoiceRow.RowAnyPartyPostOfficeBoxIdenti := '';
  InvoiceRow.RowAnyPartyOrganisationUnitNum := '';
  InvoiceRow.RowAnyPartySiteCode := '';
  InvoiceRow.INSERT;
UNTIL SalesInvoiceLine.NEXT = 0;

LOCAL InsertInvoiceRowII(SalesInvoiceLine : Record "Sales Invoice Line";VAR
InvoiceRow : Record InvoiceRow;VAR InvoiceRowII : Record "InvoiceRow II")
```

```
SalesInvoiceLine.SETRANGE("Document No.",SalesInvHeader."No.");
IF SalesInvoiceLine.FINDFIRST THEN REPEAT
  Item.GET(SalesInvoiceLine."No.");
  InvoiceRowII.INIT;
  InvoiceRowII.RowIdentifier := SalesInvoiceLine."No.";
  InvoiceRowII.RowPositionIdentifier := FORMAT(SalesInvoiceLine."Line No.");
  InvoiceRowII.RowTerminalAddressText := '';
  InvoiceRowII.RowWaybillIdentifier := '';
  InvoiceRowII.RowWaybillTypeCode := '';
  InvoiceRowII.RowClearanceIdentifier := '';
  InvoiceRowII.RowDeliveryNoteIdentifier := SalesInvoiceLine."Shipment No.";
  InvoiceRowII.RowDelivererIdentifier := SalesInvHeader."Shipping Agent Code";
  IF ShippingAgent.GET(SalesInvHeader."Shipping Agent Code") THEN
  InvoiceRowII.RowDelivererName := ShippingAgent.Name;
  InvoiceRowII.RowDelivererCountryCode := '';
  IF Country.GET(InvoiceRowII.RowDelivererCountryCode) THEN
    InvoiceRowII.RowDelivererCountryName := Country.Name;
  InvoiceRowII.RowModeOfTransportIdentifier := SalesInvoiceHeader."Shipment
Method Code";
  IF ShipmentMethod.GET(SalesInvHeader."Shipment Method Code") THEN
    InvoiceRowII.RowCarrierName := ShipmentMethod.Description;
  InvoiceRowII.RowVesselName := '';
  InvoiceRowII.RowLocationIdentifier := SalesInvoiceLine."Location Code";
  InvoiceRowII.RowTransportInformationDate := SalesInvoiceLine."Shipment
Date";//TBR
  InvoiceRowII.RowCountryOfOrigin := CompanyInfo."Country/Region Code";
//TBR
  InvoiceRowII.RowDestinationCountryCode := SalesInvHeader."Ship-to
Country/Region Code";
  IF Country.GET(SalesInvHeader."Ship-to Country/Region Code") THEN
    InvoiceRowII.RowCountryOfDestinationName := Country.Name;
  InvoiceRowII.RowPlaceOfDischarge := '';
  InvoiceRowII.RowFinalDestinationName := SalesInvHeader."Ship-to Address";
  InvoiceRowII.CNCode := '';
  InvoiceRowII.CNName := '';
  InvoiceRowII.CNOriginCountryCode := '';
  IF Country.GET(InvoiceRowII.CNOriginCountryCode) THEN
    InvoiceRowII.CNOriginCountryName := Country.Name;
  InvoiceRowII.RowManufacturerArticleIdentifi := '';
  InvoiceRowII.RowManufacturerIdentifier := '';
  InvoiceRowII.RowManufacturerName := '';
  InvoiceRowII.RowManufacturerCountryCode := '';
  IF Country.GET(InvoiceRowII.RowManufacturerCountryCode) THEN
    InvoiceRowII.RowManufacturerCountryName := Country.Name;
  InvoiceRowII.RowManufacturerOrderIdentifier := '';
  InvoiceRowII.RowPackageLength := 0;
  InvoiceRowII.RowPackageWidth := 0;
  InvoiceRowII.RowPackageHeight := 0;
  InvoiceRowII.RowPackageWeight := SalesInvoiceLine."Gross Weight";
  InvoiceRowII.RowPackageNetWeight := SalesInvoiceLine."Net Weight";
  InvoiceRowII.RowPackageVolume := SalesInvoiceLine."Unit Volume";
  InvoiceRowII.RowTransportCarriageQuantity := 0;
  InvoiceRowII.RowShortProposedAccountIdentif := '';
  InvoiceRowII.RowNormalProposedAccountIdenti := '';
  InvoiceRowII.RowProposedAccountText := '';
```

```
  InvoiceRowII.RowAccountDimensionText := '';
  InvoiceRowII.RowSellerAccountText := '';
  InvoiceRowII.RowFreeText := '';
  InvoiceRowII.RowUsedQuantity := SalesInvoiceLine.Quantity;
  InvoiceRowII.RowPreviousMeterReadingDate := 0D;
  InvoiceRowII.RowLatestMeterReadingDate := 0D;
  InvoiceRowII.RowCalculatedQuantity := SalesInvoiceLine.Quantity;
  InvoiceRowII.RowAveragePriceAmount := SalesInvoiceLine."Unit Price";
  InvoiceRowII.RowDiscountPercent := SalesInvoiceLine."Line Discount %";
  InvoiceRowII.RowDiscountAmount := SalesInvoiceLine."Line Discount Amount";
  InvoiceRowII.RowDiscountTypeCode := '';
  InvoiceRowII.RowDiscountTypeText := '';
  InvoiceRowII.RowVatRatePercent := SalesInvoiceLine."VAT %";
  InvoiceRowII.RowVatCode := SalesInvoiceLine."VAT Identifier";
  InvoiceRowII.RowVatAmount := SalesInvoiceLine."Amount Including VAT" -
SalesInvoiceLine.Amount;
  InvoiceRowII.RowVatExcludedAmount := SalesInvoiceLine.Amount;
  InvoiceRowII.RowAmount := SalesInvoiceLine."Amount Including VAT";
  //************************Might need reconstruction BEGIN
*********************************
  IF SalesInvHeader."Currency Code" <> '' THEN BEGIN
    CurrencyExchangeRate.SETRANGE(CurrencyExchangeRate."Currency Code",
SalesInvHeader."Currency Code");
    CurrencyExchangeRate.SETFILTER(CurrencyExchangeRate."Starting
Date",'<=%1', SalesInvHeader."Posting Date");
    CurrencyExchangeRate.FINDLAST;
  END;
  //************************Might need reconstruction BEGIN
*********************************
  IF SalesInvHeader."Currency Code" = '' THEN
    InvoiceRowII.OtherCurrencyAmount := SalesInvoiceLine."Amount Including
VAT"
  ELSE
    InvoiceRowII.OtherCurrencyAmount := SalesInvoiceLine."Amount Including
VAT" * CurrencyExchangeRate."Exchange Rate Amount";
  InvoiceRowII.ExchangeRate := CurrencyExchangeRate."Exchange Rate
Amount";
  InvoiceRowII.ExchangeDate := CurrencyExchangeRate."Starting Date";
  InvoiceRowII.INSERT;
UNTIL SalesInvoiceLine.NEXT = 0;
```